Flottenzuweisung in der Flugplanung

Modelle, Komplexität und Lösungsverfahren

DISSERTATION

zur Erlangung des akademischen Grades
doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Fakultät für Elektrotechnik, Informatik und Mathematik
Universität Paderborn

von
Herr Dipl.-Inform. Sven Grothklags

Dekan der
Fakultät für Elektrotechnik, Informatik und Mathematik:
Prof. Dr.-Ing. Klaus Meerkötter

Gutachter:

1. Prof. Dr. Burkhard Monien
2. Prof. Dr. Leena Suhl

Paderborn, im Oktober 2006

Danksagungen

Die vorliegende Arbeit ist während meiner Tätigkeit als wissenschaftlicher Mitarbeiter an der Universität Paderborn entstanden. Ich möchte mich an dieser Stelle bei allen bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Mein besonderer Dank gilt Prof. Dr. Burkhard Monien für die Betreuung der Arbeit. Während der letzten Jahre hat er mich stets motiviert, bei meiner Forschung unterstützt und in seiner Arbeitsgruppe eine hervorragende und fruchtbare Forschungsumgebung geschaffen.

Ich möchte mich bei allen Kollegen in der Arbeitsgruppe und dem Paderborn Center for Parallel Computing (PC^2) für die gute Zusammenarbeit und die vielen interessanten Diskussionen bedanken. Ausdrücklich nennen möchte ich die Kollegen, mit denen ich eng wissenschaftlich zusammengearbeitet habe: Torsten Fahle, Silvia Götz, Georg Kliewer, Ulf Lorenz, Thomas Sauerwald und Meinolf Sellmann. Nicht vergessen möchte ich Stefan Tschöke, der mich zur Optimierung geführt hat und mir mit seiner Energie und Kreativität stets ein Vorbild sein wird.

Die Arbeit wurde im Rahmen eines von der Deutschen Forschungsgemeinschaft geförderten Projekts im Schwerpunktprogramm „Algorithmik großer und komplexer Netzwerke“ und einer langjährigen Industriekooperation mit der Firma Lufthansa Systems durchgeführt. Daher gilt auch den Frankfurter und Berliner Kollegen von Lufthansa Systems mein Dank, besonders Georg Bolz, Jan Ehrhoff, Klaus-Peter Keilmann, Michael Lehfeld und Klaus Weber für die ausdauernde Unterstützung der Paderborner Forschung im Bereich der Flugplanung, ohne die diese Dissertation so nicht zustande gekommen wäre.

Ich möchte mich auch bei Prof. Dr. Leena Suhl für die Begutachtung meiner Dissertation bedanken. Zusätzlicher Dank gilt Ulf Lorenz und Ulf-Peter Schroeder für das Korrekturlesen der Arbeit und ihre fruchtbaren Kommentare.

Schließlich möchte ich meiner Familie und meinen Freunden für ihre unermüdliche Unterstützung und Geduld während der letzten Monate danken.

Vielen herzlichen Dank!

Paderborn, im Oktober 2006 Sven Grothklags

Inhaltsverzeichnis

Symbolverzeichnis

Symbol	Beschreibung	Seite
$\oplus_{\mathcal{T}}$	Additionsoperator auf zyklischer Planungsperiode	10
$\ominus_{\mathcal{T}}$	Subtraktionsoperator auf zyklischer Planungsperiode	10
$\Delta_{l,f}$	Flugzeugverbrauch eines Legs	59
$\Delta_{l,m,f}$	Flugzeugverbrauch einer Legverbindung	20
$\rho(t_1, t_2)$	Indikator für „wrap-around"-Intervall	20
$A_{l,f}$	mögliche Nachfolgerlegs (zyklisch)	20
$\bar{A}_{l,f}$	mögliche Nachfolgerlegs (azyklisch)	24
$B_{l,f}$	mögliche Vorgängerlegs (zyklisch)	20
$\bar{B}_{l,f}$	mögliche Vorgängerlegs (azyklisch)	24
b_l	flottenunabhängige Blockzeit	17
$b_{l,f}$	flottenabhängige Blockzeit	16
$\mathcal{F}$	Menge der Flotten/Flugzeugtypen	16
$\mathcal{F}_l$	mögliche Flotten für ein Leg	16
$\hat{f}$	virtuelle Flugzeugflotte	103
FAP(p, t, f, s)	(deterministische) Flottenzuweisungsproblemklasse	28
$g_{l,f}$	flottenabhängige Mindestbodenzeit	17
$g_{l,m,f}$	verbindungsabhängige Mindestbodenzeit	16
$g^*_{l,m,f}$	tatsächliche verbindungsabhängige Bodenzeit	20
$\mathcal{L}$	Menge der Legs	16
N_f	Anzahl verfügbarer Flugzeuge	16
$N^b_{s,f}$	zu Planungsbeginn auf Flughafen wartende Flugzeuge	23
$N^e_{s,f}$	am Planungsende Anzahl auf Flughafen wartende Flugzeuge	23
$\tilde{n}_f$	zu Periodenbeginn fliegende Flugzeuge	32
$\bar{n}^f_s$	zu Periodenbeginn auf einem Flughafen wartende Flugzeuge	32
$P(x)$	allgemeine Gewinnfunktion	16
$p_{l,f}$	Leg-Flotten-abhängiger Gewinn	18
$p_{l,m,f}$	verbindungsabhängiger Gewinn	18
$\mathcal{R}$	stochastischer Quantor	141
$\mathcal{S}$	Menge der Flughäfen	16
s^a_l	Zielflughafen	16
s^d_l	Startflughafen	16

Symbol	Beschreibung	Seite
SFAP(f)	stochastische Flottenzuweisungsproblemklasse	148
SFAPfeas(f)	stochastische Flottenzuweisungsproblemklasse	142
SFAPfeas$'(f)$	stochastische Flottenzuweisungsproblemklasse	146
$\mathcal{T}$	Länge der Planungsperiode	16
$[\mathcal{T}]$	Planungsperiode	10
$t^a_{l,f}$	flottenabhängige Ankunftszeit	21
$t^a_{l,m,f}$	verbindungsabhängige Ankunftszeit	20
t^d_l	flottenunabhängige Startzeit	16
$t^d_{l,f}$	flottenabhängige Startzeit	16
V^Δ_f	frühste Ereignisse einer Flotte	59
v^+	Nachfolgeereignis	57
v^-	Vorgängerereignis	57
$\mathcal{L}^a_v$	zu einem Ereignis landende Legs	57
$\mathcal{L}^d_v$	zu einem Ereignis startende Legs	57
$\lvert W^f_s\rvert$	Anzahl 0-Zonen einer Wartefunktion	65
W^f_s	Wartefunktion	32
$w_{s,f}$	Hilfsvariable im azyklischen Connection Network	25
$x_{l,m,f}$	Entscheidungsvariable im Connection Network	21
$y_{l,f}$	Variable für Flugkante im Time Space Network	59
$y^a_{l,f}$	Variable für ankommende Flugkante im Hybriden Modell	89
$y^d_{l,f}$	Variable für startende Flugkante im Hybriden Modell	89
z_{v,v^+}	Variable für Bodenkante im Time Space Network	59

— 1 —

Einleitung

Der Luftverkehr gehört zu den wichtigsten Verkehrsträgern weltweit und hat großen Einfluss auf die wirtschaftliche, touristische und soziale Entwicklung von Ländern und Regionen. In den letzten Jahrzehnten ist seine Bedeutung überdurchschnittlich gewachsen, wie die Statistiken der IATA (International Air Transport Association) zum Passagieraufkommen in Abbildung 1.1 zeigen. Dabei ist allerdings in den letzten fünf Jahren ein leichtes Abflachen der Zuwachsraten durch Terroranschläge, Militärkonflikte und das überregionale Ausbrechen von Krankheiten zu beobachten. Nichtsdestotrotz wird für die Zukunft ein anhaltender Anstieg des Passagier- und Frachtaufkommens im Luftverkehr erwartet.

Sinkende Wachstumsraten, hohe Treibstoffpreise und zunehmender Konkurrenzdruck gerade von so genannten Billigfluglinien haben in den letzten Jahren viele Fluglinien in die Verlustzone getrieben (Abbildung 1.1). Erste Insolvenzen und Übernahmen sind die Folge. Dabei ist die Gewinnstruktur einer Fluggesellschaft von hohen, kaum beeinflussbaren Fixkosten für die Durchführung von Flügen bei gleichzeitig schwer prognostizierbaren Erlösen aus Ticketverkäufen geprägt.

Die vielversprechenste Strategie für eine Fluggesellschaft liegt in dieser Situation vor allem darin, die Kosten des Flugbetriebs zu senken. Die Planung des Einsatzes der Ressourcen, in erster Linie der Flugzeuge und der Crews, spielt dabei die zentrale Rolle bei der Frage der Kosteneffizienz.

Die Planung bei einer Fluggesellschaft ist ein aufwendiger, mehrstufiger Prozess, der von mehreren Expertenteams durchgeführt wird, die unterschiedlichste Aufgaben lösen müssen. Erste Planungen beginnen dabei bereits ein paar Jahre vor dem Beginn einer Flugplanperiode (Sommer/Winter-Flugplan). Die Planungsteams profitieren bei ihrer Arbeit stark von der Hilfe entscheidungsunterstützender Systeme (Decision Support Systems). Diese Systeme bereiten die benötigten Informationen auf, stellen sie strukturiert zur Verfügung und regeln den Informationsaustausch zwischen den verschiedenen Planungsprozessen. Einen entscheidenden Beitrag leisten die in diesen Systemen integrierten Optimierungskomponenten. Sie

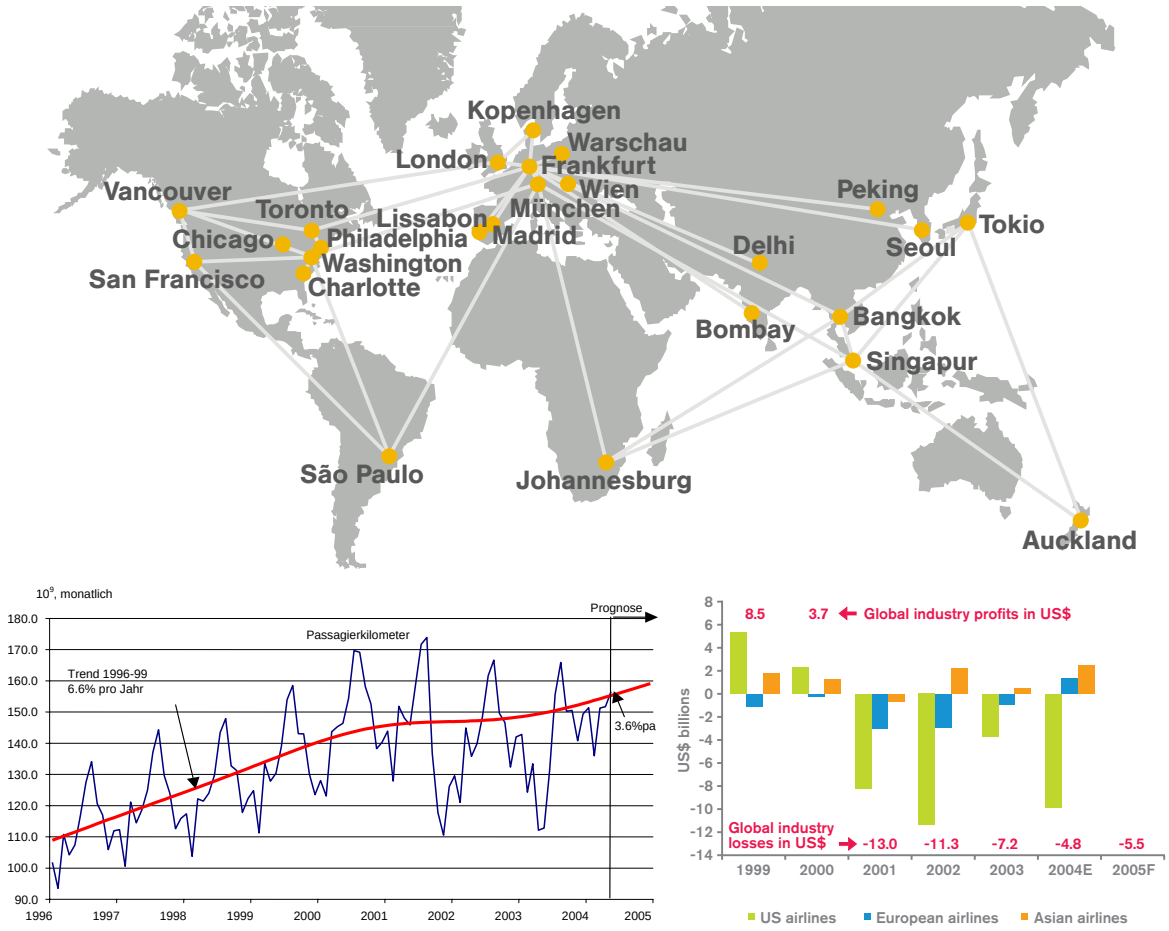


Abbildung 1.1: Oben: Beispiel für ein Internationales Flugnetzwerk; unten: Statistik zur Anzahl der Passagierkilometer und zur Entwicklung der Gewinne und Verluste in der Luftverkehrsbranche (Quelle: StarAlliance, IATA)

haben zum Ziel, kostengünstige Lösungen für komplexe Planungsszenarien, die von Experten definiert werden, zu liefern. Auf dieser Grundlage können Entscheidungen getroffen werden, die den Ressourceneinsatz entscheidend verbessern.

Eine zentrale Bedeutung in diesem Prozess fällt der Flotteneinsatzplanung einer Fluggesellschaft zu. In der mittel- und kurzfristigen Planung muss dabei der Einsatz der Flugzeugflotten für einen gegebenen Flugplan festgelegt werden. Flugzeuge verursachen mit Abstand die höchsten Fixkosten in der Bilanz einer Fluggesellschaft, sind deshalb eine knappe und teure Ressource, und ihr effizienter Einsatz ist ausschlaggebend für den Erfolg des Unternehmens. Durch die Flottenzuweisung werden zum Einen ein Großteil der anfallenden Kosten, insbesondere Treibstoff- und Crewkosten, festgelegt. Auf der anderen Seite begrenzen die unterschiedlichen Sitzkapazitäten der Flotten die maximale Anzahl an Passagieren, die auf Flugstrecken transportiert werden können, und definieren so die Transportkapazität des Flugnetzes, die mit dem prognostizierten Transportbedarf abgestimmt sein muss. Dabei kommt es in zunehmenden Maße darauf an, bei der Planung eines Teilprozesses wesentliche Aspekte nachgelagerter Prozesse zu berücksichtigen, um eine höhere Effizienz der Gesamtplanung zu

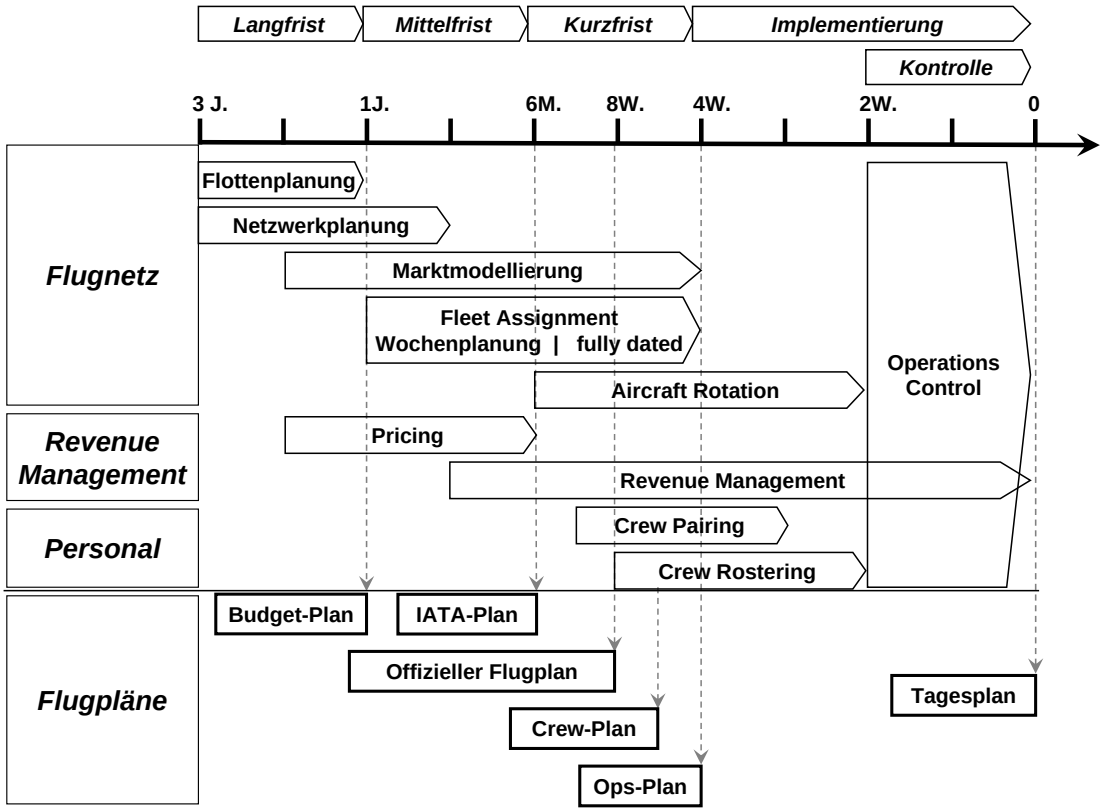


Abbildung 1.2: Übersicht über den Prozess der Flugplanung

erreichen.

1.1 Prozess der Flugplanung

In diesem Abschnitt beschreiben wir kurz die Planungsaufgaben innerhalb einer Fluggesellschaft. Ein Überblick ist in Abbildung 1.2 dargestellt. Die Grundlage für den Betrieb einer Fluggesellschaft stellt die effiziente Planung und das Management des *Flugnetzes* dar. Die daraus resultierende Transportleistung muss vom Marketing und *Revenue Management* den Flugpassagieren angeboten und verkauft werden. Schließlich erfordert die Umsetzung eines Flugplans auch den möglichst optimalen Einsatz des *Personals*: Piloten, Flugbegleiter und Bodenpersonal.

In jedem dieser Bereiche sind die unterschiedlichsten Planungen mit teils konkurrierenden Zielsetzungen durchzuführen. Erschwerend kommt hinzu, dass sich der Planungsprozess über einen sehr langen Zeitraum von mehreren Jahren erstreckt, die Ziele währenddessen variieren und zwischen den einzelnen Planungsaufgaben teils starke Abhängigkeiten existieren. Nichtsdestotrotz werden wegen der verschiedenen Planungshorizonte und der enormen Komplexität der Einzelprobleme die Planungsaufgaben meist von separaten Abteilungen getrennt voneinander durchgeführt. Allerdings gelangt man bei den Fluggesellschaften immer mehr zu

der Überzeugung, dass vor allem durch ein abgestimmtes Vorgehen zwischen den einzelnen Planungsteams eine effizientere Gesamtplanung möglich ist und damit weitere Kosteneinsparungen realisiert werden können.

In der Langfristplanung, 3 - 1 Jahr vor dem Start des Flugbetriebs, kann die Maximierung des Gewinns aus dem Betrieb des Flugnetzwerks als das wichtigste Ziel identifiziert werden. In der Mittelfristplanung (1 Jahr - 6 Monate) geht es vor allem um eine möglichst optimale Abstimmung zwischen dem Flugplan und den Ressourcen. In der Kurzfristplanung (6 Monate - 4 Wochen) wird der Einsatz der Ressourcen kontinuierlich an die Marktnachfrage angepasst und die entstehenden Kosten werden minimiert. In der Implementierungsphase (ab 4 Wochen vor dem Start) und in der Kontrollphase (ab 2 Wochen) werden Ausfälle und Sonderereignisse unter der Vorgabe der Kostenminimierung behandelt.

Als Kommunikationswerkzeug zwischen den einzelnen Planungsabteilungen, aber auch für Flugpassagiere, Reisebüros, Flugallianzpartner und Flughäfen, werden wichtige Zwischenergebnisse der Planung in unterschiedlichsten Formen weitergegeben. Im unteren Bereich der Abbildung 1.2 sind die Pläne dargestellt, die im gewissen Sinne Meilensteine in diesem Prozess bilden.

Der Budget-Plan wird 1 Jahr vor dem Start erstellt und definiert den finanziellen Rahmen des Flugplans für alle Bereiche der Fluggesellschaft. Der IATA-Plan dient als Grundlage für die halbjährliche IATA-Konferenz (Slot-Konferenz), bei der Vertreter aller Fluggesellschaften und der Flughäfen zusammenkommen, um über Verkehrs- und Landerechte (slots) zu verhandeln. Die Informationen im offiziellen Flugplan werden von Passagieren und Reisebüros für ihre Reiseplanung benutzt. Der Crew-Plan wird 6 Wochen vor dem Start erstellt und dient als Grundlage für die verbindliche Zuteilung der Ressourcen (Flugzeuge und Crews). Änderungen können ab diesem Zeitpunkt nur mit relativ hohem Aufwand vorgenommen werden. Der operative Flugplan (4 Wochen vor dem Start) wird zur Flugplanumsetzung benötigt. Kurzfristige Anpassungen sind immer notwendig und werden im Tagesverkehrsplan den betroffenen Bereichen zur Verfügung gestellt.

Im Folgenden gehen wir auf die einzelnen Planungsphasen und ihre Aufgaben für die Gesamtplanung ein.

Flottenplanung Die Neubeschaffung von Flugzeugen ist ein langwieriger Prozess. Anhand der strategischen Ziele der Fluggesellschaft, wie zum Beispiel Wachstumsstrategie, angestrebte Marktposition, Zustand der aktuellen Flugzeugflotte, und langfristiger Nachfrageprognosen wird in der Flottenplanung die Struktur der zukünftigen Flugzeugflotte festgelegt. Als Ergebnis werden Flugzeuge neu geordert oder stillgelegt, die Flughafenstruktur ausgebaut und neue Regionen als Flugziele ausgewählt.

Netzwerkplanung Die Planung umfasst in dieser Phase die Definition der anzufliegenden Flughäfen, die Anzahl der Flüge zwischen je zwei Flughäfen, die Wochentage und genauen Zeiten der Flüge und die Verknüpfung von Flügen auf großen Flughäfen (Hub-Struktur). Die Szenarien werden von Planungsabteilungen ausgearbeitet und mit Verfahren der Marktmodellierung bewertet.

Marktmodellierung Die Aufgabe der Marktmodellierung besteht darin, die Nachfrage nach der Transportleistung der Fluggesellschaft vorherzusagen. Die Grundlage bildet

eine Prognose der potentiellen Reisenden zwischen einzelnen Regionen oder Städten und wird anhand historischer Daten geschätzt. In Abhängigkeit von der Konkurrenzsituation und dem eigenen Flugnetz werden die Reisenden den eigenen Flugstrecken zugeordnet, das erwartete Transportaufkommen bewertet und die sich ergebenden Erlöse geschätzt. In dieser frühen Planungsphase stellt die Marktmodellierung das Hauptbewertungswerkzeug für die Güte anderer Planungsprozesse dar.

Flottenzuweisung (Fleet Assignment) Durch das Zuweisen von Flugzeugtypen an die einzelnen Flüge des Flugnetzes wird die Transportkapazität der Fluggesellschaft festgelegt. Die vorhandene Flotte wird möglichst kostenoptimal eingeplant, um die erwartete Nachfrage zu bedienen. Flugzeugtypen unterscheiden sich in vielen Faktoren, wie zum Beispiel in Kapazität, Reichweite und Kostenstruktur. Zu beachten sind bei der Planung diverse Restriktionen bezüglich der Flugzeuge, Flughäfen, Wartungsanforderungen usw. In der Mittelfristplanung wird auf der Basis einer repräsentativen Standardperiode (ein Tag oder eine Woche) geplant. In der Kurzfristplanung geht man auf eine konkrete Zeitperiode (fully-dated) über.

Umlaufplanung der Flugzeuge (Aircraft Rotation) Hier werden die konkreten Flugzeuge den einzelnen Strecken zugewiesen und so auch die Reihenfolge, in der Flüge nacheinander von einem Flugzeug bedient werden, festgelegt. Die resultierenden Flugzeugumläufe müssen sämtliche Wartungsregeln berücksichtigen sowie die verbindungsabhängigen Erlöse (through revenues) optimieren.

Preis- und Konditionenpolitik (Pricing) Das Ticket für einen Flug stellt kein einheitliches Gut dar. Vielmehr werden unterschiedlichen Kundengruppen (Geschäftsreisende, Freizeitreisende) differenzierte Produkte angeboten, mit unterschiedlichen Konditionen, wie zum Beispiel Stornierungsmöglichkeiten und Mindestaufenthalten. Darüber hinaus sind Ticketpreise auch vom Zeitpunkt und Ort des Kaufs (Flughafen, Reiseagentur, Internet) abhängig. Die Definition der Buchungsklassen und der zugehörigen Preise ist die Aufgabe des Pricings.

Ertragsmanagement (Revenue Management) Das Revenue Management steuert die Sitzplatzverfügbarkeit der vom Pricing festgelegten Buchungsklassen. Die betriebswirtschaftlichen Eigenschaften eines Sitzes auf einer Flugstrecke sind: niedrige variable und hohe fixe Kosten, keine Lagerfähigkeit, schwankende Nachfrage und Knappheit der Ressource. In dieser Situation muss versucht werden, jeden Platz mit einem möglichst hohen Erlös zu verkaufen. Die zentrale Frage lautet: Ist es besser, einen Platz zum jetzigen Zeitpunkt sicher mit geringen Erlösen zu verkaufen oder auf eine unsichere Nachfrage in der Zukunft mit höheren Erlösen zu warten?

Crew Pairing Die Erzeugung der Arbeitspläne für das fliegende Personal (Piloten und Flugbegleiter) ist ähnlich wie die Flugzeugeinsatzplanung zweigeteilt. Zuerst werden im Crew Pairing anonyme Arbeitspläne, die komplizierte Dienst- und Ruheregelungen respektieren müssen, erstellt. Die resultierenden Arbeitspläne (pairings) müssen dabei den Personalbedarf und die geforderten Qualifikationen der durchzuführenden Flüge abdecken und sind insbesondere von den eingesetzten Flugzeugtypen abhängig. Ziel ist es auch hier, möglichst kostengünstige Arbeitspläne zu finden, indem beispielsweise die Anzahl an Übernachtungen klein gehalten wird.

Crew Rostering Im Crew Rostering werden die konkreten Personen den vorher generierten Arbeitsplänen zugewiesen, so dass für die jeweiligen Personen gültige Dienstpläne entstehen. Dabei werden neben Abwesenheitszeiten (Urlaub, Schulungen, usw.) immer häufiger auch persönliche Präferenzen des fliegenden Personals bei der Planung berücksichtigt, um die allgemeine Personalzufriedenheit zu erhöhen.

Operations Control Beim Operations Control handelt es sich um einen Querschnittsprozess, der alle Bereiche einer Fluggesellschaft betrifft. Hauptaufgabe ist die Überwachung der Flugplanumsetzung und die Aufrechterhaltung des Flugbetriebs im Falle von Störungen (Verspätungen, Ausfällen, usw.). Dem Operations Manager stehen diverse Handlungsalternativen zur Verfügung, wie zum Beispiel das Verschieben von Flugstarts, der Einsatz von Ersatzcrews oder das Streichen von Flügen. Dabei kommt es häufig zu Umplanungen, die alle Bereiche einer Fluggesellschaft betreffen. Die Auswirkungen sollten für Passagiere möglichst gering sein, aber auch die Fluggesellschaft ist daran interessiert, möglichst schnell wieder zum „Normalbetrieb“ zurückzukehren.

Literatur über Flugplanung

Die betriebswirtschaftlichen Aspekte des Luftverkehrs werden in den beiden Lehr- und Handbüchern von [Sterzenbach and Conrady, 2003] und [Maurer, 2003] ausführlich dargestellt.

In [Barnhart et al., 2003] geben die Autoren einen Überblick über viele der in der Flugplanung auftretenden Optimierungsprobleme und identifizieren die großen Herausforderungen für die Zukunft in diesem Bereich. Auf alle in diesem Abschnitt beschriebenen Planungsprozesse wird eingegangen und der aktuelle Stand der Forschung umrissen.

In [Yu and Thengvall, 2002] und [Yu and Yang, 1998] wird ebenfalls auf die verschiedenen Optimierungsprobleme der Flugplanung eingegangen. Die grundlegenden Optimierungsmodelle werden vorgestellt und die aktuellen Lösungsmethoden beschrieben.

Der Einsatz computerunterstützter Entscheidungssysteme mit integrierten Optimierungskomponenten in der Flugplanung wird in [Suhl, 1995] behandelt.

In [Klabjan, 2005] geht der Autor auf den Einsatz von Column-Generation-Techniken in der Flugplanung, speziell in den Bereichen Crew-, Flotten- und Frachtplanung, ein.

In seiner Dissertation untersucht [Kliewer, 2005] den Prozess der Netzwerkplanung und gibt einen allgemeinen Überblick über die sonstigen Optimierungsprobleme bei Fluggesellschaften.

1.2 Ziele der Arbeit

Die Ziele der vorliegenden Arbeit liegen in einem tieferen Verständnis der Flottenzuweisung in der Flugplanung und in der Entwicklung von *praxistauglichen* Algorithmen zur Lösung solcher Optimierungsprobleme. Insbesondere geht es dabei um die Integration verschiedener Planungsphasen wie dem Ertragsmanagement oder dem Operations Control mit der Flottenzuweisung.

Daraus ergeben sich die folgenden Unterziele:

- Untersuchung der algorithmischen Komplexität von Flottenzuweisungsproblemen mit verschiedenen Ausprägungen und Erweiterungen
- Entwicklung und Evaluierung von effizienten, exakten und heuristischen Lösungsverfahren für das Grundproblem
- Identifikation von Schwachstellen bzw. wünschenswerten Erweiterungen des Grundmodells, insbesondere im Hinblick auf die Berücksichtigung von Anforderungen anderer Planungsphasen
- Entwicklung neuer Algorithmen und Modelle für obige Erweiterungen

1.3 Aufbau der Arbeit

Nach der Einleitung wird in Kapitel 2 das zentrale Optimierungsproblem dieser Arbeit, die Flottenzuweisung im Flugverkehr, formal eingeführt. Nach einer Literaturübersicht zum Stand der Forschung werden die Eingabedaten spezifiziert und zwei Varianten der Flottenzuweisung anhand ganzzahliger linearer Programme (IPs) definiert. Die erste Variante arbeitet auf einem zyklischen Planungszeitraum und wird hauptsächlich in der mittelfristigen Planung eingesetzt, die zweite Variante arbeitet auf einem konkreten (azyklischen) Planungszeitraum und findet ihre Anwendung in der kurzfristigen Planung.

Im Kapitel 3 behandeln wir die algorithmische Komplexität von Flottenzuweisungsproblemen. Zuerst wird das allgemeine Flottenzuweisungsproblem anhand von relevanten Merkmalen wie zyklische/azyklische Variante, Flottenanzahl, operationelle Eigenschaften der Flotten usw. klassifiziert und es werden die bekannten Ergebnisse in diesem Bereich vorgestellt. Es folgen neue Erkenntnisse zu Erweiterungen von 1-Flotten-Zuweisungsproblemen und insbesondere für azyklische Flottenzuweisungsprobleme. Das Highlight stellt der in Abschnitt 3.4 präsentierte Beweis der strengen NP-Vollständigkeit des Flottenzuweisungsproblems für bereits zwei Flotten dar.

In Kapitel 4 werden verschiedene selbstentwickelte exakte und heuristische Lösungsverfahren für das Flottenzuweisungsproblem vorgestellt. Nach einer Beschreibung von elementaren Transformationen, die azyklische in zyklische Flottenzuweisungsprobleme konvertieren können, und der Präsentation eines etablierten IP-Modells, des Time Space Networks, folgt die Beschreibung von neuen Lokale Suche Heuristiken für das Flottenzuweisungsproblem, einem Hill Climbing und einem Simulated Annealing Verfahren. Das Hauptaugenmerk wird dabei auf die verwendete problemspezifische Nachbarschaft gelegt, die es erlaubt, schnell *zulässige* Nachbarlösungen zu finden. In den Abschnitten 4.5 und 4.6 werden wichtige praxisrelevante Erweiterungen des Grundmodells eingeführt, die sich aus Anforderungen anderer Planungsprozesse ergeben, und wird beschrieben, wie sich die Lokale Suche Heuristiken daran anpassen lassen und wie sich neue IP-Modelle, die Grundlage für exakte Lösungsverfahren sind, dafür entwerfen lassen. Anschließend beschreiben wir verschiedene Preprocessing-Techniken, um

die Eingaben von Flottenzuweisungsproblemen aufzubereiten, ihre Zulässigkeit abzuschätzen und die Eingabegröße zu verkleinern. Am Ende wird die Performance der vorgestellten Verfahren und Techniken mittels Realdaten evaluiert.

Das stochastische Flottenzuweisungsproblem wird in Kapitel 5 eingeführt, motiviert und definiert. Dabei handelt es sich um eine Erweiterung des azyklischen Flottenzuweisungsproblems, bei der Flüge mit vorgegebenen Wahrscheinlichkeiten „gestört“ werden, was dem Verhalten bei der Umsetzung des Flugplans entspricht. Es ergibt sich ein mehrstufiges Entscheidungsproblem unter Unsicherheit, von dem wir zeigen können, dass es PSPACE-vollständig ist. Wir beschreiben einen neuen Lösungsansatz für derartige Probleme, der auf einer Modellierung als spezielles 2-Personen-Spiel aufsetzt und eine heuristische Spielbaumbewertung verwendet. Abschließende experimentelle Ergebnisse zeigen die Überlegenheit des Ansatzes gegenüber deterministischen Verfahren im Störungsmanagement/Operations Control.

In Kapitel 6 beschreiben wir verschiedene mögliche Integrationen der Phasen Marktmodellierung bzw. Ertragsmanagement und Flottenzuweisung. Nach einer Vorstellung der beiden Planungsphasen werden die Schwachstellen bei der klassischen Bewertung von Flottenzuweisungen identifiziert und das Potential einer integrierten Planung untersucht. Es werden mehrere Integrationsstrategien vorgestellt, die verschiedene Unzulänglichkeiten der klassischen Lösungsbewertung in der Flottenzuweisung beheben. Die durchgeführten Experimente belegen das Potential dieser integrierten Planung.

Im Kapitel 7 werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick für die weitere Forschung gegeben.

1.4 Ausgewählte Publikationen

Die Ergebnisse dieser Arbeit wurden in folgenden Publikationen veröffentlicht:

Begutachtete Beiträge:

- Ehrhoff, J., Grothklags, S., and Lorenz, U. (2005). Parallelism for perturbation management and robust plans. In *Proc. 11th International Euro-Par Conference (EUROPAR 2005)*, volume 3648 of *LNCS*, pages 1265–1274, Lisbon, Portugal.

- Ehrhoff, J., Grothklags, S., and Lorenz, U. (2005). Das Reparaturspiel als Formalisierung von Planung unter Zufallseinflüssen, angewendet in der Flugplanung. In Günther, H.-O., Mattfeld, D. C., and Suhl, L., editors, *Supply Chain Management und Logistik: Optimierung, Simulation, Decision Support*, pages 337–358. Physica Verlag, Heidelberg.

- Grothklags, S. (2003). Fleet assignment with connection dependent ground times. In Battista, G. D. and Zwick, U., editors, *Proc. 11th Annual European Symposium on Algorithms (ESA 2003)*, volume 2832 of *LNCS*, pages 667–678, Budapest, Hungary.

- Ehrhoff, J., Grothklags, S., Halbsgut, J., Lorenz, U., and Sauerwald, T. (2003). Robust plans and disruption management in aircraft scheduling with the help of game tree search. In *Proc. 43rd Annual Symposium of the Airline Group of the International Federation of Operational Research Societies (AGIFORS 2003)*, Paris, France.

- Weber, K., Sun, J., Sun, Z., Kliewer, G., Grothklags, S., and Jung, N. (2003). Systems integration for revenue-creating control processes. *Journal of Revenue and Pricing Management*, 2:120–137.

- Grothklags, S., Kliewer, G., and Weber, K. (2002). Improving revenue by system integration and co-operative optimization. In *Proc. 42nd Annual Symposium of the Airline Group of the International Federation of Operational Research Societies (AGIFORS 2002)*, Honolulu, USA.

Sonstige Beiträge:

- Grothklags, S., Lorenz, U., and Sauerwald, T. (2006). On the computational complexity of multi-stage decision making under uncertainty for aircraft scheduling. In *Operations Research (OR 2006)*, Karlsruhe, Germany.

- Grothklags, S., Lorenz, U., and Sauerwald, T. (2006). Stochastic airline fleet assignment is PSPACE-complete. In *5th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2006)*, Lambrecht, Germany.

- Grothklags, S., Lorenz, U., and Sauerwald, T. (2004). Experiments with the repair game. In *Aviation Application Cluster at Institute for Operations Research and Management Science (INFORMS)*, Denver, USA.

- Fahle, T., Feldmann, R., Götz, S., Grothklags, S., and Monien, B. (2003). The aircraft sequencing problem. In Klein, R., Six, H.-W., and Wegner, L. M., editors, *Computer Science in Perspective*, volume 2598 of *LNCS*, pages 152–166. Springer.

- Götz, S., Grothklags, S., Kliewer, G., and Tschöke, S. (1999). Solving the weekly fleet assignment problem for large airlines. In *Proceedings of the Third Metaheuristics International Conference (MIC 1999)*, pages 241–246, Angra dos Reis, Brasil.

1.5 Notationen

In der Arbeit werden die folgenden Notationen verwendet:

- $\mathbb{N}$ bezeichnet die Menge der natürlichen Zahlen ohne die Null
- $\mathbb{N}_0$ bezeichnet die Menge der natürlichen Zahlen inklusive der Null
- $\mathbb{Z}$ bezeichnet die Menge der ganzen Zahlen

- $\mathbb{R}$ bezeichnet die Menge der reellen Zahlen
- $[k] := \{0, 1, \ldots, k-1\}$ steht für die Menge der ersten k Zahlen aus $\mathbb{N}_0$ ($k \in \mathbb{N}$)
- $\oplus_k$ steht für den Additionsoperator auf der Restklasse $[k]$, $k \in \mathbb{Z}$; d.h. für $a, b \in \mathbb{N}_0$ gilt $a \oplus_k b := (a + b) \mod k$
- $\ominus_k$ steht für den Subtraktionsoperator auf der Restklasse $[k]$, $k \in \mathbb{Z}$; d.h. für $a, b \in \mathbb{N}_0$ gilt $a \ominus_k b := (a - b) \mod k$
- P steht für die Klasse von Entscheidungsproblemen, die sich in polynomieller Zeit von deterministischen Turing-Maschinen lösen lassen
- NP steht für die Klasse von Entscheidungsproblemen, die sich in polynomieller Zeit von nicht-deterministischen Turing-Maschinen lösen lassen
- PSPACE steht für die Klasse von Entscheidungsproblemen, die sich mit polynomiellem Platz von Turing-Maschinen lösen lassen

1.6 Grundlegende Literatur

In dieser Arbeit werden unterschiedliche algorithmische Konzepte und Optimierungsverfahren vorgestellt und untersucht. Zum grundlegenden Verständnis der benutzten Konzepte der linearen (ganzzahligen) Optimierung, Flussalgorithmen, Lokale Suche Heuristiken, Spielbaumsuche, Komplexitätstheorie usw. verweisen wir auf die folgende Literatur:

- R. K. Ahuja, T. L. Magnati, and J.B. Orlin. Network Flows: Theory, Algorithms, and Applications. Prentice Hall, 1993.
- T. H. Cormen, C. E. Leiserson, and R. L. Rivest. Introduction to Algorithms. The MIT Press, 1990.
- M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman, 1979.
- G. L. Nemhauser and L. A. Wolsey. Integer and Combinatorial Optimization. Wiley, 1988.
- G. L. Nemhauser, A. H. G. Rinnooy Kan, M. J. Todd. Optimization. Elsevier Science, 1989.
- C. H. Papadimitriou and K. Steiglitz. Combinatorial Optimization: Algorithms and Complexity. Dover Publications, 1998.
- L. A. Wolsey. Integer Programming. Wiley, 1998.

— 2 —

Das Flottenzuweisungsproblem

In diesem Kapitel definieren wir das zentrale Optimierungsproblem dieser Arbeit, das Flottenzuweisungsproblem. Zuerst geben wir eine Literaturübersicht zum Stand der Forschung auf diesem Gebiet. Anschließend führen wir das Flottenzuweisungsproblem ein, präsentieren seine Eingabedaten und definieren mit Hilfe von IP-Modellen die zwei gebräuchlichen Varianten des Flottenzuweisungsproblems, zyklische und azyklische.

2.1 Literaturübersicht

Das Problem der Flottenzuweisung im Flugverkehr beschäftigt Forscher seit über 50 Jahren. Die ersten Arbeiten stammen von [Ferguson and Dantzig, 1956]. Die von ihnen entwickelten Modelle bildeten bereits die Zuweisung von Flugzeugen auf Flüge ab und eröffneten eine jahrzehntelange intensive Forschung auf diesem Gebiet. Frühe Ansätze für das Fleet Assignment Problem (FAP) arbeiteten mit heuristischen Varianten des Frank-Wolfe Algorithmus und nutzten die Netzwerk-Struktur des Problems aus [Soumis et al., 1980]. Im Übersichtsartikel von [Etschmaier and Mathaisel, 1984] werden die frühen Arbeiten beschrieben.

Die existierenden Arbeiten lassen sich grob in drei Bereiche unterteilen. Zum einen wird das Flottenzuweisungsproblem in Form eines (gemischt-)ganzzahligen linearen Programms (IP) modelliert und mit bekannten Verfahren aus der ganzzahligen linearen Optimierung gelöst. Typischerweise kommt dabei ein Branch&Bound-Verfahren zum Einsatz, wobei in den Knoten jeweils LP-Relaxierungen des Problems berechnet werden. Wir nennen diesen Bereich „*exakte Verfahren*“, da sie zumindest potentiell in der Lage sind, optimale Flottenzuweisungen zu berechnen, wenn auch häufig die Branch&Bound-Suche heuristisch beschleunigt wird. Der andere Bereich beschreibt *heuristische Lösungsverfahren* für die Flottenzuweisung, wobei hier zumeinst Lokale Suche-Ansätze verfolgt werden. In letzter Zeit rücken verstärkt Aspekte

der *Erweiterung* des Grundmodells bzw. der Integration mit anderen Planungsphasen in den Fokus der Forschung.

Daneben haben [Gu et al., 1994] die Komplexität des FAP untersucht. Das Hauptergebnis ihrer Arbeit ist, dass das Flottenzuweisungsproblem ab drei Flugzeugtypen NP-vollständig ist.

Exakte Verfahren Mit [Abara, 1989] erschien die erste Arbeit, die ein exaktes mathematisches Modell in Form eines ganzzahligen linearen Programms für das Flottenzuweisungsproblem definierte. Abara formuliert das FAP als ein erweitertes Mehrgüter-Flussproblem, dem das so genannte Connection Network zugrunde liegt. Die entstehenden IP-Modelle lassen sich für kleinere Instanzen mit Hilfe von Branch&Bound-gestützten IP-Lösern exakt lösen.

[Daskin and Panayotopoulos, 1989] verwenden einen auf Lagrange-Relaxation basierenden Ansatz für das FAP, bei dem Routen, die von einem Hub ausgehen, mehrere Zwischenflughäfen besuchen und zum Hub zurückkehren, Flugzeugtypen zugewiesen werden.

Das bis heute am häufigsten in der Praxis eingesetzte Modell zur Flottenzuweisung stammt von [Hane et al., 1995]. Ähnlich wie Abara formulieren sie das FAP als ein erweitertes Mehrgüter-Flussproblem. Das zugrunde liegende Netzwerk, das Time Space Network, hat aber den Vorteil, mit im Vergleich zum Connection Network erheblich weniger Kanten auszukommen, so dass auch große Instanzen gelöst werden können. Im Rahmen des Branch&Bound Algorithmus werden die LP-Probleme in jedem Knoten mit einem Interior-Point Algorithmus oder einem Steepest-Edge Simplex-Algorithmus gelöst. Einen sehr wichtigen Bestandteil bilden problemspezifische Reduktionsverfahren, wie zum Beispiel die Inselbildung oder die Wahl der Branching-Variablen.

Das Time Space Network Modell bildete die Basis für viele Folgearbeiten im Bereich der Flottenzuweisung. [Berge and Hopperstad, 1993] stellen verschiedene heuristische Preprocessing-Techniken für das Time Space Network Modell vor und modifizieren das Modell, um auf Schwankungen der Passagiernachfrage kurzfristig reagieren zu können. Das erste Mal wird die Flottenzuweisung in [Desaulniers and Desrosiers, 1997] auf Wochenbasis durchgeführt. Zuvor waren nur Tagesprobleme betrachtet worden.

Die wissenschaftlichen Erkenntnisse sind dabei auch sehr schnell von der Flugindustrie mit Erfolg in die Praxis umgesetzt worden. Insbesondere durch den Einsatz kommerzieller IP-Löser wie zum Beispiel CPLEX konnten zuverlässige Optimierungssysteme für die Flottenzuweisung etabliert werden. Die Ergebnisse von [Abara, 1989] wurden in Zusammenarbeit mit American Airlines erreicht. [Hane et al., 1995] entwickelten ihre Modelle und Lösungsverfahren für Delta Air Lines. [Subramanian et al., 1994] berichten über ihre Erfahrungen mit den großen FAP-Modellen bei Delta Air Lines. Die Arbeiten bei USAir in diesem Bereich werden in [Rushmeier and Kontogiorgis, 1997] dargestellt.

Die Diplomarbeit von [Nitschke, 1997] vergleicht mehrere exakte Verfahren für das Problem der Flottenzuweisung.

Heuristische Verfahren Die oben erwähnten exakten Verfahren können potentiell die bearbeiteten Flottenzuweisungsprobleme optimal lösen und liefern im Fall, dass sie vorzeitig

beendet werden, um akzeptable Laufzeiten zu garantieren, zumindest eine belastbare Aussage über die erzielte Lösungsqualität in Form einer oberen Schranke auf den optimalen Gewinn. Da im Flottenzuweisungsproblem gerade auf der Erlösseite mit Schätzwerten gearbeitet wird, reicht eine nahezu optimale Lösung in der Praxis aus.

Heuristische Ansätze verzichten generell darauf, den Beweis der Optimalität zu führen, und lassen im Allgemeinen auch keine verlässlichen Aussagen über die erreichte Lösungsqualität zu. Ihnen geht es vielmehr darum, in kurzer Zeit möglichst gute zulässige Lösungen zu liefern.

Heuristische Verfahren für die Flottenzuweisung modifizieren zumeist existierende Zuweisungen, um bessere Lösungen zu finden. Von den bereits erwähnten Arbeiten stellen zum Beispiel [Berge and Hopperstad, 1993] Heuristiken, die mehrfach ein Minimum Cost Flow Problem lösen, oder die DELPRO-Methode, die die Zuweisung von zwei Flugsequenzen tauscht, vor. Neuere Arbeiten, die ebenfalls fertige FAP-Lösungen anpassen, sind [Jarrah et al., 2000], [Talluri, 1996], [Sharma et al., 2000] und [Yan and Young, 1996].

[Radicke, 1994] beschreibt in seiner Dissertation verschiedene heuristische Algorithmen für das FAP, die billigste Kreise in speziell konstruierten Graphen suchen und die Zuweisung der Flugzeugtypen auf diesen Kreisen verändern. Zunächst wird der Algorithmus für zwei Flugzeugtypen vorgestellt und dann auf mehrere Flotten erweitert.

In Paderborn wurden im Rahmen von mehreren Projekten heuristische Verfahren für das FAP entwickelt. In der Diplomarbeit von [Grothklags, 2000] sind Simulated Annealing Algorithmen vorgestellt worden, die auf einer komplexen, aber effizienten Nachbarschaft aufbauen. Die neueren Entwicklungen zu diesen Verfahren werden in Kapitel 4 vorgestellt.

Sosnowska präsentiert in [Sosnowska, 1999], [Sosnowska and Rolim, 2000] Simulated Annealing und GRASP-Algorithmen, die auf kleineren Flugplänen getestet werden. Dabei kommen Nachbarschaften ähnlich denen von Radicke zum Einsatz.

Erweiterungen In der Arbeit von [Rexing, 1997] werden die Abflugzeiten von Flügen nicht mehr als fest angenommen, sondern es werden alternative Abflugzeiten definiert. Dieser neue Freiheitsgrad erlaubt es, die Struktur eines Flugplans leicht zu verändern, um bessere Anschlüsse für Flugzeuge zu ermöglichen. Die Festlegung der Startzeit eines Fluges ist klassischerweise eigentlich Aufgabe der Netzwerkplanung.

In Kooperation mit dem Georgia Institute of Technology hat die amerikanische Firma SABRE Inc., Marktführer für Softwarelösungen bei Fluggesellschaften, ein so genanntes O&D Fleet Assignment System entwickelt [Jacobs and Günther, 2000], [Jacobs et al., 2000]. Das mathematische Modell für das FAP, ein Time Space Network Modell, wird periodisch mit approximierten Werten für die erzielbaren Erlöse auf Flügen im Netzwerk aktualisiert. Es werden allerdings keine Ergebnisse über die Laufzeiten oder die Qualität der berechneten Lösungen veröffentlicht. Eine ähnliche Zielsetzung verfolgen wir mit unseren Arbeiten in Kapitel 6.

Die Dissertation von [Lohatepanont, 2002] beschreibt zwei Modelle für Itinerary Based Fleet Assignment Probleme (IFAM). Im zweiten Modell wird zusätzlich noch der Netzwerkentwurf integriert. IFAM Modelle versuchen die Gewinnberechnung zu verbessern, indem Passagierflüsse im Flugnetzwerk nachgebildet werden und zur Erlösbestimmung herangezogen werden,

um so genannte Netzwerkeffekte zu erfassen. Beide Modelle haben mit ihrer enormen Größe selbst für kleine Instanzen zu kämpfen und können nur durch den Einsatz aggressiver heuristischer Preprocessing-Techniken Lösungen liefern.

[Barnhart et al., 1998] stellen so genannte Flight String Modelle für das Flottenzuweisungsproblem vor, um einen wesentlichen Aspekt der Umlaufplanung, regelmäßige Wartungsereignisse, in die Flottenzuweisung zu integrieren. Hier werden nicht mehr Flugzeuge an einzelne Flüge sondern an Flugsequenzen zugewiesen, die die geforderten Wartungsrestriktionen erfüllen. Wegen der exponentiellen Anzahl an möglichen Flugsequenzen müssen diese in einem Branch&Price-Prozess dynamisch generiert werden. Trotzdem lassen die langen Laufzeiten nur die Bearbeitung kleiner bis mittlerer Instanzen zu.

Die Arbeit von [Shenoi, 1996] befasst sich mit der Integration der Flottenzuweisung mit dem Crew Scheduling. [Cohn and Barnhart, 2003] und [Klabjan et al., 2002] berücksichtigen neben dem Crew Pairing auch noch die Umlaufplanung während der Flottenzuweisung.

[Li et al., 2006] beschreiben in ihrer Arbeit eine Integration der Flottenzuweisung mit dem Frachtrouting. Sie entwickeln ein integriertes IP-Modell, das sie mittels Benders-Dekomposition lösen.

2.2 Problemdefinition

Beim *Flottenzuweisungsproblem in der Flugplanung* (airline fleet assignment problem; FAP) muss für einen vorgegebenen Flugplan bestimmt werden, welche der vorhandenen Flugzeugflotten (oder auch Flugzeugtypen) welchen Flug übernehmen soll. Der *Flugplan* besteht dabei aus einer Menge von so genannten *Legs* (Non-Stop-Flüge zwischen zwei Flughäfen), die jeweils durch ihren Start- und Zielflughafen sowie ihre Abflugzeit definiert sind.

Bei der Zuweisung eines Flugzeugtyps an ein Leg können eine ganze Reihe von Faktoren einen Einfluss auf die Flottenauswahl haben:

- erwartetes Passagieraufkommen auf dem Leg
- Sitzplatzangebot eines Flugzeugtyps
- erwartete Erlöse
- Treibstoffverbrauch
- benötigte Crew-Größe
- Flugdistanz des Legs
- maximale Reichweite eines Flugzeugtyps
- wichtige Anschlussflüge eines Legs
- maximale Geschwindigkeit eines Flugzeugtyps

- Wartungsmöglichkeiten am Start- bzw. Zielflughafen
- Verfügbarkeit zum Flugzeugtyp passender Gates am Start- bzw. Zielflughafen
- Lärmschutzauflagen am Start- bzw. Zielflughafen
- Lautstärke eines Flugzeugtyps
- ...

Die ersten fünf Punkte haben einen direkten Einfluss auf den Gewinn, der sich mit einer Zuweisung erzielen lässt, und werden deshalb typischerweise in Form einer zu optimierenden Zielfunktion berücksichtigt.

Die darauf folgenden Punkte beschreiben eher Beschränkungen an die Flotten, die einem Leg zugewiesen werden dürfen. Es ist zum Beispiel offensichtlich, das ein Langstreckenflug nicht von einem Flugzeug mit zu kurzer Reichweite geflogen werden darf. Hierbei handelt es sich um eine unbedingt einzuhaltende Einschränkung. Andererseits geben wichtige Anschlussflüge eine spätestmögliche Landezeit für ein Leg vor, damit die Passagiere genügend Zeit zum Umsteigen haben. Zu langsame Flugzeugtypen können hier zwar prinzipiell eingesetzt, sollten dann aber in der Zielfunktion zusätzlich bestraft werden.

Neben den bisher erwähnten Aspekten werden an eine zulässige Lösung für das Flottenzuweisungsproblem noch die Forderungen gestellt, dass jedes Leg des Flugplans von genau einer Flugzeugflotte geflogen wird und dass die Lösung mit der verfügbaren Anzahl an Flugzeugen der einzelnen Flotten auskommt. Diese Bedingungen machen das Flottenzuweisungsproblem im Allgemeinen NP-schwer (Kapitel 3).

Somit handelt es sich bei dem Flottenzuweisungsproblem in der Flugplanung um ein kombinatorisch schweres *Optimierungsproblem*, das aus der Menge von zulässigen Flottenzuweisungen diejenige auswählen soll, die den höchsten Gesamtgewinn erzielt.

In dieser Arbeit unterscheiden wir zwischen zwei Varianten des Flottenzuweisungsproblems, zyklischen und azyklischen.

Bei der zyklischen Flottenzuweisung wird der Planungszeitraum als zyklisch angesehen. Das heißt, dass das Ende der Planungsperiode nahtlos an den Beginn der Planungsperiode anschließt. Hier kann es zum Beispiel passieren, dass ein Leg, das zu einem Zeitpunkt nahe des Endes der Planungsperiode startet, eine kleinere Lande- als Startzeit besitzt. Die zyklische Flottenzuweisung kommt hauptsächlich in der Mittelfristplanung zum Einsatz.

Bei der azyklischen Flottenzuweisung kann der Planungszeitraum als unbeschränkt angesehen werden. Eine Konsequenz daraus ist, dass die Landezeiten von Legs immer größer als ihre Startzeiten sind. Azyklische Flottenzuweisungsprobleme treten typischerweise in der Kurzfristplanung und im Störungsmanagement auf.

In der Literatur ist bisher fast ausschließlich die zyklische Variante untersucht worden, obwohl beide Varianten in der Praxis von großer Bedeutung sind und teilweise Unterschiede in ihrer algorithmischen Komplexität aufweisen (Kapitel 3).

2.2.1 Eingabedaten

Definition 2.1 (Eingabeparameter). *In ihrer allgemeinsten Form besteht eine Instanz des Flottenzuweisungsproblems aus den folgenden Eingabedaten:*

$\mathcal{T}$	$\in \mathbb{N}$; *Länge der Planungsperiode*
$\mathcal{F}$	*Menge der Flotten*
N_f	$\in \mathbb{N}$ *Anzahl verfügbarer Flugzeuge von Flotte* $f \in \mathcal{F}$
$\mathcal{L}$	*Menge der Legs*
$\mathcal{S}$	*Menge der Flughäfen, die von Legs aus* $\mathcal{L}$ *benutzt werden*
$\mathcal{F}_l$	$\subseteq \mathcal{F}$ *mit* $\mathcal{F}_l \neq \emptyset$; *Menge der Flotten, die Leg* $l \in \mathcal{L}$ *fliegen können*
s_l^d	$\in \mathcal{S}$; *Startflughafen von Leg* $l \in \mathcal{L}$
s_l^a	$\in \mathcal{S}$; *Zielflughafen von Leg* $l \in \mathcal{L}$
$t_{l,f}^d$	$\in [\mathcal{T}]$; *Startzeitpunkt von Leg* $l \in \mathcal{L}$, *wenn es von einem Flugzeug der Flotte* $f \in \mathcal{F}_l$ *geflogen wird*
$b_{l,f}$	$\in \mathbb{N}$; *Blockzeit von Leg* $l \in \mathcal{L}$, *wenn es von einem Flugzeug der Flotte* $f \in \mathcal{F}_l$ *geflogen wird*
$g_{l,m,f}$	$\in \mathbb{N}_0$; *Mindestbodenzeit (auf Flughafen* s_l^a*), wenn ein Flugzeug der Flotte* $f \in \mathcal{F}_l$ *im Anschluss an Leg* $l \in \mathcal{L}$ *das Leg* $m \in \mathcal{L}$ *fliegt*
P	*Gewinnfunktion, die zu einem zulässigen Fleet Assignment den resultierenden Gewinn berechnet*

Die benötigten Eingabedaten für ein zyklisches bzw. azyklisches Flottenzuweisungsproblem unterscheiden sich praktisch nicht voneinander. Im azyklischen Fall kann die Periondenlänge $\mathcal{T}$ als unendlich angesehen werden, da hier die Länge der Planungsperiode implizit durch die Start-, Block- und Mindestbodenzeiten der Legs definiert wird.

Die Menge $\mathcal{F}_l$ gibt für jedes Leg an, welche Flugzeugtypen dem Leg zugewiesen werden dürfen. Unter anderem können operationelle Restriktionen (Reichweite, Lärmschutz, ...) die Menge der möglichen Flotten für ein Leg einschränken. Da in einer zulässigen Flottenzuweisung jedem Leg genau ein Flugzeugtyp zugewiesen werden muss, ist eine notwendige Bedingung für die Existenz einer zulässigen Lösung, dass alle $\mathcal{F}_l$ mindestens eine Flotte enthalten.

Abhängig von der eingesetzten Flotte gibt $t_{l,f}^d$ den Startzeitpunkt eines Legs an. Meistens ist der Startzeitpunkt in der Praxis unabhängig von der eingesetzten Flotte, da dieser auf den internationalen Slot-Konferenzen zwischen den Fluggesellschaften weit vor der eigentlichen Flotteneinsatzplanung ausgehandelt wird. Ist der Startzeitpunkt unabhängig von der eingesetzten Flotte, bezeichnen wir ihn auch einfach mit t_l^d.

$b_{l,f}$ bezeichnet die Dauer eines Legs in Abhängigkeit von der eingesetzten Flotte. Diese Dauer umfasst neben der eigentlichen Zeit, während der sich das Flugzeug in der Luft befindet, auch die Zeit auf dem Start- und Zielflughafen, die das Flugzeug zwischen Gate/Terminal und Start- bzw. Landebahn benötigt. $b_{l,f}$ wird in der Flugplanung Blockzeit genannt.[1]

[1] *Blocks* sind die Bremsklötze, die während der Standzeit eines Flugzeugs auf einem Flughafen um die Bereifung gelegt werden, und die Blockzeit bezeichnet die Zeit, während der ein Flugzeug unterwegs ist (off-blocks).

Vereinfachend wird in der Literatur teilweise angenommen, dass die Blockzeit unabhängig von der verwendeten Flugzeugflotte ist. In einem solchen Fall bezeichnen wir die Blockzeit auch einfach mit b_l. Diese Annahme ist in der Praxis aber höchstens bei Kurzstreckenflügen realistisch und kann dazu führen, dass bereits durch die Modellierung der erzielbare Gewinn geschmälert wird.

Ein großes Problem im Zusammenhang mit der Blockzeit stellt die Tatsache dar, dass es sich bei der Blockzeit eigentlich nicht um eine feste, bekannte Zahl handelt. In Abhängigkeit von der Verkehrsdichte, dem Wetter, usw. schwankt die tatsächliche Blockzeit eines Legs von Mal zu Mal, und die Blockzeit sollte eher als eine Zufallsverteilung aufgefasst werden. In der traditionellen Flugplanung wird nichtsdestotrotz eine feste Dauer angenommen, z.B. der Mittelwert oder eine Dauer, die in 70% der Fälle ausreicht. Welche Konsequenzen sich aus einer genaueren Berücksichtigung von zufallsverteilten Blockzeiten ergeben, ist Thema von Kapitel 5.

Nach der Landung von Leg l auf dem Flughafen s_l^a kann ein Flugzeug vom Typ f als nächstes einen weiteren Flug m durchführen, der von eben diesem Flughafen startet ($s_m^d = s_l^a$). Allerdings kann es dies nicht direkt im Anschluss an seine Landezeit $t_{l,f}^d \oplus_{\mathcal{T}} b_{l,f}$ tun, sondern es muss erst eine gewisse Zeit am Terminal warten, damit die Passagiere aus- und einsteigen können, das Gepäck ent- und beladen werden kann, das Flugzeug neu betankt werden kann, usw. Die Zeit, die für diese Handlungen mindestens eingeplant werden muss, wird Mindestbodenzeit $g_{l,m,f}$ genannt.

In der hier eingeführten allgemeinen Form handelt es sich bei $g_{l,m,f}$ um eine *verbindungsabhängige* Mindestbodenzeit, da sie, neben dem Flugzeugtyp, sowohl von dem landenden als auch startenden Leg abhängt. Bisher ist in der Literatur fast ausschließlich der Fall betrachtet worden, dass die Mindestbodenzeit nur von dem landenden Leg (oder, noch spezieller, dem Flughafen) und dem Flugzeugtyp abhängig ist. Wir nennen diese Art von Mindestbodenzeiten *flottenabhängig* und bezeichnen sie mit $g_{l,f}$. In diesem Fall kann die Mindestbodenzeit ohne Beschränkung der Allgemeinheit der (ebenfalls flottenabhängigen) Blockzeit $t_{l,f}^d$ zugeschlagen werden und in der weiteren Planung als Null bzw. nicht relevant angenommen werden.

In der Praxis treten aber häufig Fälle auf, bei der die Mindestbodenzeit eben auch vom Folgeleg abhängig ist. Zum Beispiel kann die Mindestbodenzeit zwischen zwei Langstreckenflügen um bis zu 30 Minuten länger sein als zwischen einem Langstreckenflug und einem Kurzstreckenflug, da unter anderem mehr Gepäck ent- und beladen werden muss. Die Zeitunterschiede sind in diesem Fall allerdings im Allgemeinen nicht besonders groß.

Gravierender fallen die Zeitunterschiede bei verbindungsabhängigen Mindestbodenzeiten in folgendem Fall aus: Ein Flughafen besitzt separate Terminals für Inlands- und Auslandsflüge. Das Auslandsterminal stellt zusätzliche Einrichtungen für Einreise- und Zollformalitäten bereit. Wenn nun das Flugzeug eines ankommenden Auslandsflugs als nächstes einen Inlandsflug absolvieren soll, muss es vom Auslandsterminal zum Inlandsterminal mit Hilfe eines Schleppers gezogen werden, was deutlich über eine Stunde dauern kann. Würde das Flugzeug als nächstes einen weiteren Auslandflug durchführen, könnte es am Auslandsterminal bleiben und die Mindestbodenzeit wäre entsprechend kürzer.

Streng genommen handelt es sich bei verbindungsabhängigen Mindestbodenzeiten um An-

forderungen, die erst in der auf die Flottenzuweisung anschließenden Planungsphase der Rotationsbildung berücksichtigt werden müssten, da es *nicht* die Aufgabe der Flottenzuweisung ist, die Folgebeziehungen von Legs festzulegen. Allerdings führt eine Nichtberücksichtigung von verbindungsabhängigen Mindestbodenzeiten während der Flottenzuweisung unter Umständen dazu, dass während der Rotationsbildung keine zulässigen Lösungen gefunden werden können. Mit verbindungsabhängigen Mindestbodenzeiten werden somit Teilaspekte einer nachgelagerten Planungsaufgabe in die Flottenzuweisung integriert.

Die Bewertung einer zulässigen Flottenzuweisung erfordert, wenn sie möglichst genau erfolgen soll, in der Praxis eine aufwendige Berechnung. Kapitel 6 geht näher darauf ein. Wir nehmen hier erst einmal an, dass diese Berechnung von einer Funktion P durchgeführt wird, die als Eingabe eine (beliebig kodierte) zulässige Lösung übergeben bekommt und als Ergebnis den Gewinn dieser Lösung liefert.

Als vereinfachende Näherung wird traditionell in die Flottenzuweisung angenommen, dass der Gewinn, den ein Leg l abwirft, nur von der dem Leg l zugewiesenen Flotte f abhängt: Unterschiedliche Flugzeugtypen verursachen unterschiedliche Kosten, wenn sie ein Leg fliegen, und können unterschiedlich viele Passagiere mitnehmen und damit unterschiedlich viel Erlös erwirtschaften. Sogenannte Netzwerkeffekte (siehe Kapitel 6) bleiben dabei unberücksichtigt.

Eine etwas genauere Gewinnberechnung lässt sich durchführen, wenn man neben Leg-Flotten-abhängigen Gewinnen auch so genannte verbindungsabhängige Gewinne berücksichtigt. Zum Beispiel verursacht das weiter vorne erwähnte Schleppen eines Flugzeugs von einem Auslands- zu einem Inlandsterminal zusätzliche Kosten, die mit verbindungsabhängigen Gewinnen modelliert werden können. Ferner ist es so, dass sich die Attraktivität von Flugverbindungen erhöht, wenn man bei einem Zwischenstopp das Flugzeug nicht wechseln muss. Es sollten also bevorzugt solche Flüge nacheinander von einem Flugzeug bedient werden, die von vielen Passagieren als Flugverbindung genutzt werden. Dies erhöht das Passagieraufkommen und erlaubt höhere Ticketpreise. Ähnlich wie bei den verbindungsabhängigen Bodenzeiten werden verbindungsabhängige Gewinne traditionell eigentlich erst während der Rotationsbildung berücksichtigt und optimiert.

Definition 2.2 (Alternative Eingabeparameter). *Alternativ zu der Gewinnfunktion P können als Eingabedaten Leg-Flotten- und verbindungsabhängige Gewinne angegeben werden:*

$p_{l,f}$	$\in \mathbb{Z}$; *Gewinn, den Leg $l \in \mathcal{L}$ abwirft, wenn es von Flotte $f \in \mathcal{F}$ bedient wird*
$p_{l,m,f}$	$\in \mathbb{Z}$; *(zusätzlicher) Gewinn, wenn ein Flugzeug der Flotte $f \in \mathcal{F}$ nacheinander die Legs $l \in \mathcal{L}$ und $m \in \mathcal{L}$ bedient*

Die Eingabe einer Flottenzuweisungsinstanz lässt sich normalerweise mit $O(|\mathcal{F}| \cdot |\mathcal{L}|)$ Zahlen kodieren. Dies gilt offensichtlich, wenn keine verbindungsabhängigen Mindestbodenzeiten und Gewinne berücksichtigt werden müssen. Verbindungsabhängige Angaben können im worstcase $O(|\mathcal{F}| \cdot |\mathcal{L}|^2)$ Zahlen erfordern. Allerdings lassen sie sich meistens sehr einfach effizienter Kodieren.[2]

[2] Zum Beispiel durch Weglassen von Null-Werten.

2.2.2 Das zyklische Flottenzuweisungsproblem

Das zyklische Flottenzuweisungsproblem wird überwiegend in der mittelfristigen, strategischen Planung eingesetzt, wo für einen repräsentativen Zeitraum (typischerweise ein Tag oder eine Woche) der Flotteneinsatz geplant werden soll.

Eine zulässige Lösung muss hier auf ein beliebiges Vielfaches des Planungszeitraums ausgedehnt werden können, indem einfach entsprechend viele Lösungskopien aneinandergereiht werden. Dies ist gleichbedeutend mit der Forderung, dass in einer zulässigen Lösung für jede Flotte auf jedem Flughafen die Anzahl an von dieser Flotte bedienten startenden Legs gleich der Anzahl an landenden Legs ist. Man sagt in diesem Zusammenhang auch, dass die Flughäfen bezüglich der Starts und Landungen jeder Flotte *balanciert* sein müssen. Eine notwendige Bedingung für die Existenz einer solchen Lösung ist offensichtlich, dass bereits im Eingabeflugplan die Anzahl von startenden mit der Anzahl von landenden Legs auf jedem Flughafen übereinstimmt.

Eines der ersten Modelle, das für das zyklische Flottenzuweisungsproblem entwickelt wurde, stammt von [Abara, 1989]. Es handelt sich um ein ganzzahliges lineares Programm (IP), das als ein erweitertes ganzzahliges (Mehrgüter-)Fluss-Problem angesehen werden kann. Das zugrunde liegende gerichtete Flussnetzwerk $G = (V, E)$, *Connection Network* genannt, ist wie folgt definiert:

$$V = \{(l, f) \mid l \in \mathcal{L} \wedge f \in \mathcal{F}_l\} \subseteq \mathcal{L} \times \mathcal{F}$$
$$E = \left\{((l, f), (m, f)) \mid (l, f), (m, f) \in V \wedge s_l^a = s_m^d\right\},$$

wobei die Kantenkapazitäten alle 1 sind. Es gibt keine expliziten Quellen oder Senken im Netzwerk.

Die Knoten des Netzwerks bestehen aus Leg-Flotten-Tupeln. Für jede erlaubte Zuweisung einer Flotte an ein Leg existiert ein Knoten. Zwischen zwei Knoten existiert genau dann eine Kante, wenn die zugehörigen Legs l und m nacheinander von einem Flugzeug bedient werden können. Dafür muss zum einen die Flotte beider Knoten gleich sein, da ein Flugzeug nicht seine Flottenzugehörigkeit ändern kann. Zum anderen muss der Zielflughafen von Leg l mit dem Startflughafen von Leg m übereinstimmen. Das heißt, es sind keine Überführungsflüge (ferry flights) zwischen verschiedenen Flughäfen zugelassen. Nach dieser Konstruktion besteht das Connection Network aus $|\mathcal{F}|$ viele unabhängigen Teilnetzwerken, da nie Knoten, die mit unterschiedlichen Flotten assoziiert sind, miteinander durch eine Kante verbunden sind.

Es ist anzumerken, dass bei der Entscheidung, ob zwei Legs nacheinander von einem Flugzeug bedient werden können, die Start- und Landezeiten *keine* Rolle spielen. Die Zyklizität der Planungsperiode erlaubt es, dass ein Flugzeug nach seiner Landung *jedes* vom Landeflughafen startende Leg erreichen kann, also insbesondere auch Legs, deren Startzeit vor der eigenen Landezeit liegt.

Die Güter, die im Connection Network verschickt werden können, repräsentieren die Flugzeuge der einzelnen Flotten. Ein Fluss von 1 auf einer Kante $((l, f), (m, f))$ steht für ein Flugzeug der Flotte f, das erst Leg l und anschließend Leg m bedient. Der Durchfluss durch einen Knoten (l, f) gibt damit die Anzahl der Flugzeuge von Typ f an, die Leg l bedienen.

An einen Fluss im Connection Network, der eine zulässige Flottenzuweisung repräsentiert, werden die folgenden erweiterten Anforderungen gestellt:

- Jedes Leg muss von genau einem Flugzeug bedient werden, das heißt, der Gesamtdurchfluss durch alle Knoten, die ein Leg l repräsentieren, muss genau 1 sein. Hierdurch werden die aus Graphensicht unabhängigen Teilnetzwerke des Connection Networks miteinander gekoppelt.

- Die Flusserhaltung in jedem Knoten garantiert, dass ein Flugzeug nach der Landung immer ein passendes Anschlussleg besitzt. In Abwesenheit von Quellen und Senken können so nur zirkuläre Flüsse im Netzwerk entstehen, und es ist garantiert, dass die Flughäfen je Flotte bezüglich Starts und Landungen balanciert sind.

- Der Fluss muss mit der vorgegebenen Flugzeuganzahl je Flotte auskommen. Ein Weg, die von einem Fluss benötigte Flugzeuganzahl einer Flotte f zu bestimmen, ist, einen Schnitt durch das Netzwerk „zu einem festen Zeitpunkt t“ zu legen und den Fluss über diesen Schnitt zu bestimmen. t kann im Prinzip beliebig innerhalb der Planungperiode gewählt werden. Besonders einfach ist es aber, den Periodenbeginn zu wählen.

All diese Bedingungen lassen sich einfach in ein ganzzahliges lineares Programm übersetzten. Wir definieren dazu:

$A_{l,f}$ $= \{m \in \mathcal{L} \mid s^d_m = s^a_l \wedge f \in \mathcal{F}_m\}$
Menge der Legs, die von einem Flugzeug der Flotte $f \in \mathcal{F}_l$ direkt im Anschluss an Leg $l \in \mathcal{L}$ geflogen werden können

$B_{l,f}$ $= \{k \in \mathcal{L} \mid s^a_k = s^d_l \wedge f \in \mathcal{F}_k\}$
Menge der Legs, die von einem Flugzeug der Flotte $f \in \mathcal{F}_l$ direkt vor Leg $l \in \mathcal{L}$ geflogen werden können

$\rho(t_1, t_2)$ $= \begin{cases} 1 & \text{, falls } t_1 > t_2 \\ 0 & \text{, sonst} \end{cases}$
Indikatorfunktion, die angibt, ob das offene Intervall (t_1, t_2) in der zyklischen Gruppe $[\mathcal{T}]$ den Wert Null (Periodenbeginn) enthält oder nicht

$t^a_{l,m,f}$ $= t^d_{l,f} \oplus_{\mathcal{T}} b_{l,f} \oplus_{\mathcal{T}} g_{l,m,f}$
Ankunftszeit (ready time) genannter Zeitpunkt, ab dem ein Flugzeug von Flotte f nach der Landung von Leg l bereit ist, Leg m als nächstes zu bedienen

$g^*_{l,m,f}$ $= g_{l,m,f} + (t^d_{m,f} \ominus_{\mathcal{T}} t^a_{l,m,f})$ tatsächliche Bodenzeit, wenn die Legs l und m nacheinander von einem Flugzeug der Flotte f bedient werden

$\Delta_{l,m,f}$ $= \left\lfloor \frac{b_{l,f} + g^*_{l,m,f}}{\mathcal{T}} \right\rfloor + \rho(t^d_{l,f}, t^d_{m,f})$
Anzahl Flugzeuge von Flotte f, die benötigt werden, wenn die Legs l und m nacheinander von Flotte f bedient werden

Die Mengen $A_{l,f}$ bzw. $B_{l,f}$ repräsentieren die direkten Nachfolger bzw. Vorgänger eines Knotens (l, f) im Connection Netzwerk. $\Delta_{l,m,f}$ beschreibt, wie oft eine Kante zwischen den Knoten (l, f) und (m, f) den Periodenbeginn „schneidet“ und entspricht damit der Anzahl an Flugzeugen von Flotte f, die ein Fluss von 1 auf dieser Kante verbraucht. Es wird dabei

angenommen, dass eine Kante $((l, f), (m, f))$ den Zeitraum vom Start des Legs l bis zum Start des Folgelegs m repräsentiert.

Die Funktion $\rho(t_1, t_2)$ bestimmt, wie oft das Intervall (t_1, t_2) den Periodenanfang enthält.[3] Da die Dauer einer Kante im Connection Network (Blockzeit + Mindestbodenzeit + zusätzliche Bodenzeit bis zum nächsten Abflug) größer als die Periodendauer sein kann, ist es möglich, dass eine Kante mehr als ein Flugzeug verbraucht. Der erste Term in der Definition von $\Delta_{l,m,f}$ bestimmt, wie viele Flugzeuge allein durch die Block- und Bodenzeit verbraucht werden. Der zweite Term bestimmt, ob zusätzlich noch zwischen dem Abflug von Leg l und dem Weiterflug mit Leg m ein Periodenbeginn liegt.

Bemerkung 2.3. *Der Begriff* Ankunftszeit *für $t^a_{l,m,f}$ bezeichnet hier nicht den Zeitpunkt, zu dem ein Flugzeug am Terminal ankommt, sondern den Zeitpunkt, ab dem es seinen nächsten Flug absolvieren kann. Im Falle von verbindungsabhängigen Mindestbodenzeiten hängt dieser Zeitpunkt auch vom Folgeleg m ab.*

Liegen keine verbindungsabhängigen Mindestbodenzeiten vor, schreiben wir für die Ankunftszeit auch einfach $t^a_{l,f}$.

Das Modell verwendet nur eine Klasse von Variablen, die den Fluss auf den Kanten des Connection Networks repräsentieren:

$x_{l,m,f}$ Boolsche Variable, die anzeigt, ob die Legs l und m unmittelbar nacheinander von einem Flugzeug der Flotte $f \in \mathcal{F}$ bedient werden sollen ($x_{l,m,f} = 1$) oder nicht ($x_{l,m,f} = 0$)

Das zyklische Flottenzuweisungsproblem lässt sich damit wie folgt formulieren:

Modell 2.4 (Zyklisches Connection Network).

$$\textit{Maximiere } P(x) \tag{2.1}$$

unter den Nebenbedingungen

$$\sum_{f \in \mathcal{F}_l} \sum_{m \in A_{l,f}} x_{l,m,f} = 1 \qquad \forall l \in \mathcal{L} \tag{2.2}$$

$$\sum_{k \in B_{l,f}} x_{k,l,f} - \sum_{m \in A_{l,f}} x_{l,m,f} = 0 \qquad \forall l \in \mathcal{L}, f \in \mathcal{F}_l \tag{2.3}$$

$$\sum_{l \in \mathcal{L}: f \in \mathcal{F}_l} \sum_{m \in A_{l,f}} \Delta_{l,m,f} x_{l,m,f} \leq N_f \qquad \forall f \in \mathcal{F} \tag{2.4}$$

$$x_{l,m,f} \in \{0, 1\} \qquad \forall l \in \mathcal{L}, f \in \mathcal{F}_l, m \in A_{l,f} \tag{2.5}$$

Die Gleichungen (2.2) bestimmen für jedes Leg l den Gesamtdurchfluss durch seine zugehörigen Knoten im Connection Network und legen diesen Wert auf 1 fest. Damit wird, wie vorne erwähnt, jedes Leg von genau einer Flotte bedient. Der Durchfluss durch einen Knoten wird dabei einfach mit Hilfe seines Ausgangsflusses bestimmt.[4]

[3] Dies kann entweder gar nicht der Fall sein ($t_1 \leq t_2$) oder einmal ($t_1 > t_2$).

[4] Alternativ kann natürlich auch der Eingangsfluss in einen Knoten verwendet werden.

Die Gleichungen (2.3) sind die normalen Flusserhaltungsgleichungen. Da im Connection Network keine Quellen und Senken existieren, ist die rechte Seite immer gleich Null. Hierdurch wird sicher gestellt, dass sich die Flottenzuweisung eines Legs l nicht während des Flugs ändern kann: Wenn ein Flugzeug von Flotte f im Anschluss an Leg l ein Leg m bedient ($x_{l,m,f} = 1$), so muss es vorher ein passendes Leg k bedient haben ($x_{k,l,f} = 1$).

Die Ungleichungen (2.4) beschränken für jede Flotte die Anzahl verwendeter Flugzeuge. Dazu wird für jede Flotte der Gesamtfluss „über einen Schnitt zum Zeitpunkt Null“ bestimmt.

Die Bedingungen (2.5) definieren die verwendeten Variablen als Boolsch. Wegen der im Vergleich zu einem normalen Fluss-Problem zusätzlichen Bedingungen (2.2) und (2.4) kann ansonsten nicht garantiert werden, dass ein ganzzahliger Fluss gefunden wird.

Schließlich legt die Zielfunktion (2.1) fest, dass wir unter allen zulässigen Flottenzuweisungen eine mit maximalem Gewinn suchen.

Ist die Gewinnfunktion in Form von Leg-Flotten- und verbindungsabhängigen Gewinnen gegeben, kann die Zielfunktion (2.1) wie folgt ersetzt werden, und wir erhalten eine Formulierung als ganzzahliges lineares Programm:

$$\text{Maximiere} \sum_{l \in \mathcal{L}} \sum_{f \in \mathcal{F}_l} \sum_{m \in A_{l,f}} (p_{l,f} + p_{l,m,f}) x_{l,m,f} \tag{2.6}$$

Aus der Variablenbelegung einer zulässigen Lösung von Modell 2.4 lässt sich praktisch direkt die Zuweisung der Flotten an die einzelnen Legs ablesen und somit eine Lösung für das Flottenzuweisungsproblem bestimmen. Darüber hinaus gibt das Modell zusätzlich explizit an, in welcher Reihenfolge die Legs des Flugplans von den Flugzeugen der einzelnen Flotten bedient werden sollen. Dies ist mehr als von einem klassischen Flottenzuweisungsproblem verlangt wird und eigentlich Teil der nachgelagerten Planungsphase der Umlaufgenerierung. Allerdings können so einfach verbindungsabhängige Mindestbodenzeit und Gewinne bereits während der Flottenzuweisung berücksichtigt werden.

Das Connection Network Modell 2.4 ist damit eine sehr ausdrucksstarke Methode, Flottenzuweisungsprobleme zu modellieren und zu lösen. Es hat allerdings einen gravierenden Nachteil, der das Modell in der Praxis bereits für mittelgroße Probleminstanzen unattraktiv macht: Die entstehenden ganzzahligen linearen Programme enthalten zu viele Variablen und können (zumindest mit Standardlösern) selbst auf aktuellen Computern kaum gelöst werden.

Beobachtung 2.5 (Größe des Connection Network Modells). *Eine Flottenzuweisungsinstanz mit Legmenge $\mathcal{L}$ und Flottenmenge $\mathcal{F}$ erzeugt ein Connection Network Modell mit $O(|\mathcal{F}| \cdot |\mathcal{L}|^2)$ vielen Variablen und $O(|\mathcal{F}| \cdot |\mathcal{L}|)$ Nebenbedingungen. Offensichtlich lässt sich ein entsprechendes Modell in polynomieller Zeit erzeugen.*

Beim Zählen der Nebenbedingungen haben wir die Variablendefinitionen 2.5 nicht mitberücksichtigt, da diese in IP-Lösern typischerweise nicht in Form von expliziten Nebenbedingungen sondern als Attribut der Variablen berücksichtigt werden.

Problematisch ist hier, dass die Variablenanzahl quadratisch mit der Leganzahl wachsen kann. Der schlimmste Fall tritt ein, wenn es einen Flughafen gibt, auf dem alle Legs entweder starten oder landen. Unglücklicherweise haben reale Flugnetzwerke gerade die Eigenschaft, dass fast alle Legs von einem (bzw. einigen wenigen) Flughäfen[5] starten bzw. dort landen. Reale Probleminstanzen führen somit tatsächlich zu einem Connection Network Modell mit $\Theta(|\mathcal{F}| \cdot |\mathcal{L}|^2)$ vielen Variablen.

2.2.3 Das azyklische Flottenzuweisungsproblem

Das azyklische Flottenzuweisungsproblem kommt normalerweise in der kurzfristen, taktischen Planung zum Einsatz und dient dazu, für einen konkreten Zeitraum eine Flottenzuweisung zu berechnen. Man spricht in diesem Zusammenhang auch von einer *fully-dated* Planung. Die Dauer der Planungsperiode kann je nach Einsatzzweck stark variieren, von einem halben Tag im Störungsmanagement bis hin zu 6 oder mehr Wochen in der Kurzfristplanung bei sich ändernden Flugplänen.

Im Gegensatz zur zyklischen Flottenzuweisung kann man bei der azyklischen Variante die Planungsperiode als unbeschränkt ansehen, was zur Folge hat, dass für jedes Leg seine Landezeit größer ist als seine Startzeit und ein Flugzeug nach seiner Landung als nächstes nur Legs bedienen kann, die später starten.

Desweiteren wird die Forderung fallen gelassen, dass die Lösung eines azyklischen Flottenzuweisungsproblems beliebig oft hintereinander ausgeführt werden kann. Daher müssen weder die Eingabedaten noch eine zulässige Lösung bezüglich der Starts und Landungen auf den einzelnen Flughäfen balanciert sein.

Die Anzahl verfügbarer Flugzeuge je Flotte f kann beim azyklischen Flottenzuweisungsproblem wie bei der zyklischen Variante explizit mit dem Eingabeparameter N_f vorgegeben werden. In der Praxis gebräuchlicher sind allerdings Untervarianten, bei denen die Verteilung der Flugzeuge der Flotten auf die einzelnen Flughäfen zu Beginn und/oder zum Ende der Planungsperiode vorgegeben werden. Die Gesamtanzahl verfügbarer Flugzeuge je Flotte ergibt sich dann implizit durch Aufsummieren über alle Flughäfen.

Definition 2.6 (Alternative Eingabeparameter). *Alternativ zu den Gesamtflugzeuganzahlen N_f können als Eingabedaten für das azyklische Flottenzuweisungsproblem die Verteilung der Flugzeuge zu Beginn und/oder zum Ende der Planungsperiode angegeben werden:*

$N^b_{s,f}$ $\in \mathbb{N}_0$; *Anzahl Flugzeuge von Flotte f, die zu Beginn der Planungsperiode auf dem Flughafen $s \in \mathcal{S}$ bereit stehen*

$N^e_{s,f}$ $\in \mathbb{N}_0$; *Anzahl Flugzeuge von Flotte f, die am Ende der Planungsperiode auf dem Flughafen $s \in \mathcal{S}$ verfügbar sein müssen*

Das Connection Network Modell von [Abara, 1989] lässt sich einfach an die azyklische Variante anpassen. Das wesentliche Teil $G = (V, E)$ des zugrunde liegenden Flussnetzwerk ist

[5] Solche Flughäfen werden Hubs genannt.

wie folgt definiert:

$$V = \{(l, f) \mid l \in \mathcal{L} \wedge f \in \mathcal{F}_l\} \subseteq \mathcal{L} \times \mathcal{F}$$
$$E = \left\{((l, f), (m, f)) \mid (l, f), (m, f) \in V \wedge s^a_l = s^d_m \wedge t^a_{l,m,f} \leq t^d_{m,f}\right\},$$

wobei die Kantenkapazitäten wieder alle 1 sind. Je nach der Art und Weise, wie die verfügbaren Flugzeuge je Flotte vorgegeben sind, muss dieses Teilnetzwerk noch passend um Quellen und Senken erweitert werden. Wir gehen hier nicht näher darauf ein; die Details lassen sich aber aus der folgenden Beschreibung des resultierenden ganzzahligen linearen Programms und seiner Untervarianten entnehmen.

Der Hauptunterschied (neben der Existenz von Quellen und Senken) zwischen dem zyklischen und dem azyklischen Connection Networks liegt in der Definition der Kantenmenge E. Im azyklischen Fall existiert eine Kante zwischen zwei Knoten (l, f) und (m, f) nur dann, wenn zusätzlich zu den Bedingungen der zyklischen Variante die Ankunftszeit von Leg l nicht hinter der Startzeit von Leg m liegt, da *ein* Flugzeug eine solche Legfolge nicht bedienen könnte.

Eine Folge daraus ist, dass es sich beim dem Connection Network im azyklischen Fall um einen gerichteten azyklischen Graphen (DAG) handelt, da Kanten als in die Zukunft gerichtet angesehen werden können. Ein Fluss in dem Netzwerk lässt sich somit immer in eine Menge von Pfaden aufteilen und jeder Pfad repräsentiert dabei die Folge von Legs, die ein Flugzeug nacheinander bedienen muss. Ferner liefert die Anzahl der Pfade damit auch die insgesamt benötigte Flugzeuganzahl.

Für das lineare Programm des azyklischen Flottenzuweisungsproblems definieren wir nun:

$\bar{A}_{l,f}$ $= \{m \in \mathcal{L} \mid s^d_m = s^a_l \wedge f \in \mathcal{F}_m \wedge t^a_{l,m,f} \leq t^d_{m,f}\} \cup \{*\}$
Menge der Legs, die von einem Flugzeug der Flotte $f \in \mathcal{F}_l$ direkt im Anschluss an Leg $l \in \mathcal{L}$ geflogen werden können

$\bar{B}_{l,f}$ $= \{k \in \mathcal{L} \mid s^a_k = s^d_l \wedge f \in \mathcal{F}_k \wedge t^a_{k,l,f} \leq t^d_{l,f}\} \cup \{*\}$
Menge der Legs, die von einem Flugzeug der Flotte $f \in \mathcal{F}_l$ direkt vor Leg $l \in \mathcal{L}$ geflogen werden können

Hierbei ist $* \notin \mathcal{L}$ die Bezeichnung für ein Hilfsleg, das anzeigt, dass ein Leg kein direktes Vorgänger- bzw. Nachfolgerleg besitzt.

Das azyklische Flottenzuweisungsproblem lässt sich damit wie folgt formulieren:

Modell 2.7 (Azyklisches Connection Network).

Maximiere $P(x)$ (2.7)

unter den Nebenbedingungen

$$\sum_{f\in\mathcal{F}_l}\sum_{m\in\bar{A}_{l,f}} x_{l,m,f} = 1 \qquad \forall l \in \mathcal{L} \tag{2.8}$$

$$\sum_{k\in\bar{B}_{l,f}} x_{k,l,f} - \sum_{m\in\bar{A}_{l,f}} x_{l,m,f} = 0 \qquad \forall l \in \mathcal{L}, f \in \mathcal{F}_l \tag{2.9}$$

$$\sum_{l\in\mathcal{L}:f\in\mathcal{F}_l} x_{l,*,f} \leq N_f \qquad \forall f \in \mathcal{F} \tag{2.10}$$

$$x_{l,m,f} \in \{0,1\} \qquad \forall l \in \mathcal{L}, f \in \mathcal{F}_l, m \in A_{l,f} \tag{2.11}$$

$$x_{*,l,f} \in \{0,1\} \qquad \forall l \in \mathcal{L}, f \in \mathcal{F}_l \tag{2.12}$$

Das Modell unterscheidet sich kaum von dem zyklischen Connection Network Modell 2.4. Die Mengen $\bar{A}_{l,f}$ und $\bar{B}_{l,f}$ der möglichen Vorgänger und Nachfolger eines Legs sind entsprechend der azyklischen Struktur des Fluss-Netzwerks angepasst worden. Es existieren zusätzliche Variablen $x_{*,l,f}$ bzw. $x_{l,*,f}$[6], die Fluss von einer Quelle bzw. zu einer Senke transportieren können. Ein Fluss von 1 für solch eine Variable zeigt an, dass das zugehörige Leg l das erste bzw. das letzte Leg ist, dass von einem Flugzeug der Flotte f bedient wird. Entsprechend wird durch die Ungleichungen (2.10) die Flugzeuganzahl je Flotte f dadurch begrenzt, dass die Anzahl der Legs bestimmt wird, die als letztes von einem Flugzeug der Flotte f bedient werden.

Die Zielfunktion (2.7) lässt sich wie im zyklischen Fall linearisieren, falls nur Leg-Flotten- und verbindungsabhängige Gewinne in der Eingabe gegeben sind. Somit lässt sich dann das azyklische Flottenzuweisungsproblem ebenfalls mittels eines ganzzahligen linearen Programms beschreiben. Allerdings wächst auch hier die Variablenanzahl quadratisch mit der Leganzahl.

Modell 2.7 beschreibt den Fall, dass in der Eingabe die Flugzeuganzahlen je Flotte gegeben sind. Sind alternativ die Verteilungen der Flugzeuge zu Periodenbeginn und zum Periodenende vorgegeben, ersetzt man die Ungleichungen (2.10) durch

$$w_{s,f} + \sum_{l\in\mathcal{L}:f\in\mathcal{F}_l\wedge s_l^d=s} x_{*,l,f} = N^b_{s,f} \qquad \forall s \in \mathcal{S}, f \in \mathcal{F} \tag{2.13}$$

$$w_{s,f} + \sum_{l\in\mathcal{L}:f\in\mathcal{F}_l\wedge s_l^a=s} x_{l,*,f} = N^e_{s,f} \qquad \forall s \in \mathcal{S}, f \in \mathcal{F} \tag{2.14}$$

$$w_{s,f} \in \mathbb{N}_0 \qquad \forall s \in \mathcal{S}, f \in \mathcal{F} \tag{2.15}$$

Die Gleichungen (2.13) bestimmen für jeden Flughafen s, wie viele Flugzeuge der Flotte f dort ihre Arbeit aufnehmen, und setzen diesen Wert mit der vorgegebenen Anzahl gleich. Entsprechendes leisten die Gleichungen (2.14) für die Flugzeuge einer Flotte f, die ihren Dienst auf Flughafen s beenden. Die zusätzlichen Variablen $w_{s,f}$ unterstützen den Fall, dass Flugzeuge auf einem Flughafen während der gesamten Planungsperiode warten und währenddessen kein einziges Leg bedienen.

Sind in der Eingabe die Verteilungen entweder nur zu Periodenbeginn oder am Periodenende vorgegeben, lässt man im Modell die Gleichungen der nicht spezifizierten Grenze einfach weg.

[6] Die Variablen $x_{l,*,f}$ werden im Modell von den Bedingungen (2.11) definiert.

— 3 —

Komplexität

Die Flottenzuweisung gehört zu den besonders schweren Optimierungsproblemen. Selbst stark eingeschränkte Unterproblemklassen sind bereits streng NP-vollständig. Zuerst teilen wir daher in diesem Kapitel das Flottenzuweisungsproblem anhand relevanter Charakteristika in verschiedene Unterproblemklassen auf. Wir stellen die bisher bekannten Ergebnisse vor, die sich allesamt nur mit der zyklischen Flottenzuweisung beschäftigen. Dann untersuchen wir den Ein-Flotten-Fall genauer und zeigen, dass sich das Problem auch mit zusätzlichen Anforderungen wie verbindungsabhängigen Mindestbodenzeiten zumeist noch effizient lösen lässt. Den Hauptteil bildet die anschließende Reduktion von SAT auf den Zwei-Flotten-Fall, die zeigt, dass bereits die meisten Flottenzuweisungsprobleme mit zwei Flotten streng NP-vollständig und nicht approximierbar sind. Abschließend untersuchen wir speziell die azyklischen Problemklassen genauer. Es stellt sich heraus, das manche azyklischen Flottenzuweisungsprobleme einfacher zu lösen sind als ihre zyklischen Entsprechungen. Nichtsdestotrotz ist aber das Optimierungsproblem der azyklischen Flottenzuweisung für mehr als zwei Flotten wie im zyklischen Fall streng NP-vollständig.

In diesem Kapitel werden wir Komplexitätsergebnisse zu Entscheidungs- und Optimierungsproblemen vorstellen. Für Komplexitätsaussagen über Optimierungsprobleme gilt dabei:

Bemerkung 3.1. *Wenn von einem Optimierungsproblem gesagt wird, dass es zu NP gehört bzw. NP-vollständig ist, meinen wir, dass das* korrespondierende Entscheidungsproblem *zu NP gehört bzw. NP-vollständig ist.*

Dabei lautet beispielsweise das korrespondierende Entscheidungsproblem zu einem Maximierungsproblem

$$\text{Finde } x^* \in L \text{ mit } f(x^*) = \max_{x \in L} f(x)$$

wie folgt:

$$\text{Existiert ein } x \in L \text{ mit } f(x) \geq F?$$

3.1 Problemklassen

Für die Komplexitätbetrachtungen in diesem Kapitel benötigen wir eine detaillierte Unterscheidung von verschiedenen Entscheidungs- und Optimierungsproblemen, die im Rahmen der Flottenzuweisung auftreten können. Dabei spielen unter anderem die Flottenanzahl und die operationellen Eigenschaften der Flotten eine zentrale Rolle. Wir definieren deshalb:

Definition 3.2. *FAP*$(p, t, f, s[, e])$ *steht für die Problemklasse von Flottenzuweisungsproblemen mit den folgenden Attributen:*

- $p \in \{Feas, Inf, Opt\}$ *definiert das zu betrachtende Optimierungs- oder Entscheidungsproblem.*
  - $Feas$ *steht für das Entscheidungsproblem, ob eine zulässige Lösung existiert oder nicht.*
  - Inf *steht für ein Optimierungsproblem, bei dem es keine Beschränkung für die Flugzeuganzahl der Flotten gibt.*
  - *Schließlich steht* Opt *für das Optimierungsproblem mit beschränkter Flugzeuganzahl.*
- $t \in \{Cy, Ac, Ac0, Ac1, Ac2\}$ *legt fest, ob es sich um ein zyklisches* $(t = Cy)$ *oder azyklisches* $(t = Ac)$ *Flottenzuweisungsproblem handelt. Bei azyklischen Varianten unterscheiden wir bei Bedarf ferner, ob die Verteilung der Flugzeuge der einzelnen Flotten zum Periodenbeginn bzw. -ende gar nicht spezifiziert ist* $(t = Ac0)$*, an einer der Grenzen* $(t = Ac1)$ *oder an beiden Grenzen* $(t = Ac2)$*.* Ac *stellt somit die Vereinigung der Fälle* $Ac0$*,* $Ac1$ *und* $Ac2$ *dar.*
- $f = |\mathcal{F}| \in \mathbb{N}$ *steht für die Anzahl der verschiedenen Flotten.*
- $s \in \{Eq, Or, Ar\}$ *definiert die operationellen Eigenschaften der Flotten.*
  - Eq *steht dafür, dass alle Flotten alle Legs in gleicher Zeit fliegen können.*
  - Or *steht dafür, dass alle Flotten alle Legs fliegen können und es eine Ordnung* $f_1, \ldots, f_{|\mathcal{F}|}$ *auf den Flotten (Geschwindigkeit) gibt, so dass für alle* $l \in \mathcal{L}$ *gilt:* $i < j \Longrightarrow b_{l,f_i} \leq b_{l,f_j}$
  - Ar *steht für den allgemeinsten Fall, bei dem für jedes Leg unabhängig die erlaubten Flotten und deren Flugzeiten festgelegt sind.*
- *Optional können als letzte Attribute weitere Eigenschaften definiert werden.*
  - $e = Cdt$ *besagt, dass verbindungsabhängige Mindestbodenzeiten auftreten dürfen. Wird* Cdt *nicht angegeben, wird ohne Beschränkung der Allgemeinheit angenommen, dass alle Mindestbodenzeiten Null sind.*[1]

[1] Etwaige Mindestbodenzeiten lassen sich in diesem Fall den Blockzeiten zuschlagen (siehe Seite 17).

- $e = Cdp$ *besagt, dass verbindungsabhängige Gewinne definiert sein dürfen. Wird* Cdp *nicht angegeben, wird angenommen, dass in der Zielfunktion nur Leg-Flotten-abhängige Gewinne auftreten.*
- $e = Big$ *besagt, dass Block- und Mindestbodenzeiten größer als die Länge der Planungsperiode sein dürfen. Wird* Big *nicht angegeben, wird angenommen, dass die Blockzeit plus die anschließende Mindestbodenzeit eines jeden Legs kleiner als die Länge der Planungsperiode ist.*

In allen Fällen gehen wir davon aus, dass die Startzeiten der Legs unabhängig von der verwendeten Flotte sind und dass die zu optimierende Gewinnfunktion die Form der linearen Zielfunktion (2.6) *besitzt.*

Nicht alle möglichen Attributkombinationen führen zu sinnvollen Problemklassen. So macht es für eine Problemklasse mit $p = Inf$ keinen Sinn, die azyklischen Untervarianten $Ac1$ und $Ac2$ zu betrachten, da diese implizit die Flugzeuganzahl vorgeben. Desweiteren macht für azyklische Problemklassen die Angabe von $e = Big$ keinen Sinn, da bei azyklischen Flottenzuweisungsproblemen die Periodenlänge als unbeschränkt angesehen werden kann. Schließlich ist bei Problemklassen mit nur einer Flotte ($f = 1$) das Attribut s bedeutungslos.

Satz 3.3. *Alle oben definierten Problemklassen des Flottenzuweisungsproblems FAP*$(*)$ *gehören zu NP.*

Beweis. Alle Probleminstanzklassen FAP$(*)$ lassen sich offensichtlich mit Hilfe von ganzzahligen linearen Programmen gemäß der Modelle 2.4 und 2.7 modellieren. Für $p = Feas$ ist die Zielfunktion irrelevant, und für $p = Inf$ können die Ungleichungen (2.4) bzw. (2.10) im jeweiligen Modell entfallen.

Nach Beobachtung 2.5 lassen sich somit Instanzen der Klassen FAP$(*)$ in polynomieller Zeit auf das Problem der *ganzzahligen linearen Programmierung* reduzieren, das bekanntermaßen NP-vollständig ist [Borosh and Treybis, 1976], [Karp, 1972] und damit in NP liegt. □

Bemerkung 3.4. *Die Klassen FAP*$(Feas, *)$ *und FAP*$(Inf, *)$ *sind in der entsprechenden Klasse FAP*$(Opt, *)$ *in dem Sinne enthalten, dass ein Algorithmus für FAP*$(Opt, *)$ *auch alle Probleme aus FAP*$(Feas, *)$ *und FAP*$(Inf, *)$ *in gleicher Zeit lösen kann. Umgekehrt übertragen sich NP-Vollständigkeitsaussagen über Klassen FAP*$(Feas, *)$ *oder FAP*$(Inf, *)$ *direkt auf die entsprechende FAP*$(Opt, *)$*-Klasse.*

3.2 Bekannte Ergebnisse

Es existieren nur wenige Arbeiten, die sich mit der algorithmischen Komplexität des Flottenzuweisungsproblems beschäftigen, und es existieren nur Arbeiten zum zyklischen Flottenzuweisungsproblem.

3.2.1 Ergebnisse von Gu et al.

Die meisten bekannten Ergebnisse stammen aus der Arbeit von [Gu et al., 1994]. Darin wird die Komplexität der zyklischen Klassen FAP(p, Cy, f, Eq) für $p \in \{Feas, Inf\}$ und $f \in \mathbb{N}$ untersucht. Es handelt sich dabei um die Problemklassen mit den restriktivsten Einschränkungen an die operationellen Eigenschaften der Flotten und an die Gewinnfunktion.

Satz 3.5. *Für $p \in \{Feas, Inf\}$ gilt:*

1. *Instanzen der Klassen FAP$(p, Cy, 1, Eq)$ können in polynomieller Zeit gelöst werden.*
2. *Instanzen der Klasse FAP$(Inf, Cy, 2, Eq)$ können in polynomieller Zeit gelöst werden.*
3. *Die Klassen FAP(p, Cy, f, Eq) sind für $f \geq 3$ streng NP-vollständig.*

Näheres zu 1. folgt in Abschnitt 3.3.1. Instanzen aus FAP$(Inf, Cy, 2, Eq)$ lassen sich auf Single(!)-Commodity-Min-Cost-Flow-Probleme reduzieren und dann mit bekannten polynomiellen Algorithmen lösen [Ahuja et al., 1993]. Die Komplexität der Klassen FAP$(Feas, Cy, 2, Eq)$ und FAP$(Opt, Cy, 2, Eq)$ ist unbekannt. Auf die Klassen aus 3. lassen sich ganzzahlige Zwei-Güter-Flussprobleme reduzieren, die, wie in [Even et al., 1976] bewiesen, bekanntermaßen NP-vollständig sind. Ähnliche Reduktionen verwenden wir in Abschnitt 3.5 für azyklische Flottenzuweisungsprobleme.

Darüber hinaus wird in [Gu et al., 1994] gezeigt, dass bereits die Klasse FAP$(Feas, Cy, 1, Eq)$ streng NP-vollständig ist, wenn man von zulässigen Lösungen zusätzlich verlangt, dass alle Flugzeuge der Flotte *eine* gemeinsame Rundtour absolvieren müssen.

Eine wichtige Konsequenz aus den Ergebnissen von [Gu et al., 1994] ist:

Folgerung 3.6. *Das zyklische Flottenzuweisungsproblem FAP$(Opt, Cy, *)$ ist für mehr als zwei Flotten streng NP-vollständig. Desweiteren ist es nicht in polynomieller Zeit approximierbar, falls P $\neq$ NP ist.*

Beweis. Da die Klassen FAP(p, Cy, f, Eq) die meisten Einschränkungen an eine Flottenzuweisungsinstanz bezüglich Flotteneigenschaften und Gewinnfunktion stellen, können allgemeinere Probleme nicht einfacher sein. Nach Bemerkung 3.4 überträgt sich die strenge NP-Vollständigkeit von FAP$(Feas, *)$ und FAP$(Inf, *)$ auf FAP$(Opt, *)$. Die Nicht-Approximierbarkeit ist eine direkte Konsequenz aus der Tatsache, dass bereits FAP$(Feas, Cy, 3, Eq)$ NP-vollständig ist. □

3.2.2 Ergebnisse von Radicke

Die einzige weitere Arbeit, in der sich Ergebnisse zur Komplexität der Flottenzuweisung finden, ist die Dissertation von [Radicke, 1994]. Für den zyklischen Zwei-Flotten-Fall zeigt er:

Satz 3.7.

1. *Die Klasse FAP$(Opt, Cy, 2, Ar)$ ist streng NP-vollständig.*

2. *Die Klasse FAP$(Feas, Cy, 2, Eq, Big)$ ist NP-vollständig.*

Punkt 1. wird durch eine Reduktion von SAT bewiesen. Hierbei müssen die beiden Flotten sowohl unterschiedliche Blockzeiten für einzelne Legs besitzen können als auch manche Legs nur von einer der beiden Flotten bedient werden dürfen. Eine neue, allgemeinere Reduktion, die stärkere Einschränkungen an die Flotten erlaubt, sowohl für zyklische als auch für azyklische Flottenzuweisungsprobleme geeignet ist und die strenge NP-Vollständigkeit der FAP$(Feas, *)$-Variante zeigt, wird in Abschnitt 3.4 vorgestellt.

In einer kurzen Nebenbemerkung erwähnt Radicke, dass sich Punkt 2. durch eine Reduktion vom Subset-Sum-Problem beweisen lässt. In der Literatur werden häufig die FAP$(*, Big)$-Problemklassen nicht als Flottenzuweisungsprobleme im engeren Sinne angesehen, da sie aus Praxissicht unrealistisch sind.

3.3 Der Ein-Flotten-Fall

Bei nur einer Flotte f ist das Flottenzuweisungsproblem dahingehend trivial, dass klar ist, das jedem Leg eben diese eine Flotte zugewiesen werden muss. Ohne verbindungsabhängige Gewinne ist damit auch der erzielbare Gewinn leicht durch $\sum_{l \in \mathcal{L}} p_{l,f}$ in linearer Zeit berechenbar. Somit bleibt im Ein-Flotten-Fall (ohne verbindungsabhängige Gewinne) nur die Frage zu klären, ob es überhaupt eine zulässige Lösung gibt.

Die Hauptaufgabe besteht folglich darin zu überprüfen, ob die vorgegebene Flugzeuganzahl eingehalten wird. Wir bestimmen hier die Anzahl benötigter Flugzeuge ähnlich wie im Connection Network Modell, indem wir zählen, wo sich am Periodenanfang wie viele Flugzeuge befinden.

3.3.1 Ohne verbindungsabhängige Mindestbodenzeiten

Für Flottenzuweisungsprobleme ohne verbindungsabhängige Mindestbodenzeiten lässt sich bei gegebener Flottenzuweisung die Anzahl der auf einem Flughafen benötigten Flugzeuge mit Hilfe der so genannten Wartefunktion effizient bestimmen. Sei dazu

z $: \mathcal{L} \to \mathcal{F}$
die vorgegebene Flottenzuweisung

$nd_s^f(t)$ $= \left|\left\{l \in \mathcal{L} \mid z(l) = f \wedge s_l^d = s \wedge t_{l,f}^d \leq t\right\}\right|$
die Anzahl an Starts auf Flughafen s von Flotte f im Intervall $[0, t]$

$na_s^f(t)$ $= \left|\left\{l \in \mathcal{L} \mid z(l) = f \wedge s_l^a = s \wedge t_{l,f}^a \leq t\right\}\right|$
die Anzahl an Landungen auf Flughafen s von Flotte f im Intervall $[0, t]$

$n_s^f(t)$ $= na_s^f(t) - nd_s^f(t)$

Definition 3.8 (Wartefunktion). *Zu einer vorgegebenen Flottenzuweisung z ist die Wartefunktion $W_s^f : [\mathcal{T}] \to \mathbb{N}_0$ für einen Flughafen $s \in \mathcal{S}$ und eine Flotte $f \in \mathcal{F}$ wie folgt definiert:*

$$W_s^f(t) = n_s^f(t) + \bar{n}_s^f,$$

wobei

$$\bar{n}_s^f = -\min_{t \in [\mathcal{T}]} n_s^f(t)$$

ist.

Nach [Gu et al., 1994] gilt:

Satz 3.9. *Für eine gegebene zulässige Zuweisung z eines Flottenzuweisungsproblems ohne verbindungsabhängige Mindestbodenzeiten müssen zum Periodenbeginn auf einem Flughafen s von Flotte f mindestens $\bar{n}_s^f$ Flugzeuge warten, damit alle Starts und Landungen auf diesem Flughafen während der Planungsperiode ausgeführt werden können. Ferner reicht diese Anzahl auch aus.*

$W_s^f(t)$ beschreibt dann für jeden Zeitpunkt $t \in [\mathcal{T}]$ der Planungsperiode, wie viele Flugzeuge von Flotte f zum Zeitpunkt t auf Flughafen s warten.

Bemerkung 3.10. *Nach Definition ist der Wertebereich der Wartefunktion nicht-negativ und nimmt während der Planungsperiode mindestens einmal den Wert Null an.*

Ein Flughafen s ist bezüglich der Starts und Landungen einer Flotte f genau dann balanciert, wenn $n_s^f(\mathcal{T}-1) = 0$ gilt. Also muss eine zulässige Lösung eines zyklischen Zuweisungsproblems $n_s^f(\mathcal{T}-1) = 0$, oder äquivalent $W_s^f(\mathcal{T}-1) = \bar{n}_s^f$, für alle Flughäfen s und alle Flotten f erfüllen.

In zyklischen Flottenzuweisungsproblemen können sich zum Periodenbeginn bereits einige Flugzeuge in der Luft befinden. Wie bei der Definition des zyklischen Connection Networks in Abschnitt 2.2.2 lässt sich deren Anzahl wie folgt bestimmen:

Satz 3.11. *Bei einer zulässigen Zuweisung z eines zyklischen Flottenzuweisungsproblems ohne verbindungsabhängige Mindestbodenzeiten befinden sich zu Periodenbeginn von Flotte f*

$$\tilde{n}_f = \sum_{l \in \mathcal{L}: z(l) = f} \left\lfloor \frac{b_{l,f} + g_{l,f}}{\mathcal{T}} \right\rfloor + \rho(t_{l,f}^d, t_{l,f}^a)$$

viele Flugzeuge in der Luft.

Daraus folgt nun:

Satz 3.12. *Instanzen der Klassen FAP$(Opt, t, 1, Ar[, Big])$ mit $t \in \{Cy, Ac\}$ lassen sich in Zeit $O(|\mathcal{L}| \log |\mathcal{L}|)$ lösen.*

Beweis. Die einzig mögliche Zuweisung ist, jedes Leg mit der einen verfügbaren Flotte f zu bedienen. Deren Gewinn $\sum_{l\in\mathcal{L}} p_{l,f}$ kann in Zeit $O(|\mathcal{L}|)$ berechnet werden.

Es verbleibt zu überprüfen, ob die Zuweisung zulässig ist, das heißt, die geforderte Flugzeuganzahl einhält und, im zyklischen Fall, balanciert ist. Dazu bestimmen wir mittels der Wartefunktionen die zum Periodenbeginn benötigten Flugzeuge auf den Flughäfen. Man legt ein Array mit den Start- $(s_l^d, t_{l,f}^d)$ und Landeereignissen $(s_l^a, t_{l,f}^a)$ aller Legs l an und sortiert dieses lexikographisch nach Flughafen und Zeit. Ein anschließender linearer Scan liefert alle $\bar{n}_s^f$-Werte und überprüft, falls nötig ($t = Cy$), ob die einzelnen Flughäfen bezüglich Starts und Landungen balanciert sind. All dies lässt sich offensichtlich in Zeit $O(|\mathcal{L}| \log |\mathcal{L}|)$ durchführen.

Im zyklischen Fall befinden sich zusätzlich noch $\tilde{n}_f$ Flugzeuge in der Luft, was in Zeit $O(|\mathcal{L}|)$ berechnet werden kann.

Alle Werte aufsummiert ergeben die insgesamt von der Zuweisung benötigte Anzahl Flugzeuge von Flotte f, die mit der vorgegebenen Anzahl verglichen wird. Für den Fall, dass bei azyklischen Problemen die Verteilungen der Flugzeuge auf den Flughäfen zu Periodenbeginn und/oder zum Periodenende vorgegeben sind ($t \in \{Ac1, Ac2\}$), lässt sich die Zulässigkeit direkt an den $\bar{n}_s^f$- und $W_s^f(\mathcal{T}-1)$- Werten ablesen. □

Bemerkung 3.13. *Neben der Überprüfung, ob eine gegebene Zuweisung z für ein Flottenzuweisungsproblem ohne verbindungsabhängige Mindestbodenzeiten zulässig ist, lassen sich auch zu z passende Flugzeugumläufe in Zeit $O(|\mathcal{L}| \log |\mathcal{L}|)$ bestimmen. Dabei wird dann wie im Connection Network Modell jedem Leg l ein Nachfolgeleg m zugewiesen, das als nächstes von dem Flugzeug, das l geflogen hat, bedient wird. Ein einfaches FIFO-Verfahren auf den sortierten Start-/Landeereignissen reicht dafür aus [Gertsbach and Gurevich, 1977].*

3.3.2 Mit verbindungsabhängigen Mindestbodenzeiten

Bei Flottenzuweisungsproblemen mit verbindungsabhängigen Mindestbodenzeiten lassen sich die Flugzeuganzahlen nicht mehr so einfach mit der Wartefunktion bestimmen. Hier müssen zumeist Matching-Probleme auf bipartiten Graphen gelöst werden.

Satz 3.14. *Instanzen der zyklischen Klassen FAP$(Opt, Cy, 1, Ar, Cdt[, Big])$ lassen sich in Zeit $O(|\mathcal{L}|^3)$ lösen.*

Beweis. Das zyklische Connection Network Modell 2.4 sieht für die Klasse FAP$(Opt, Cy, 1, Ar, Cdt[, Big])$ wie folgt aus:

$$\text{Maximiere} \sum_{l\in\mathcal{L}} \sum_{m\in A_{l,f}} p_{l,f} x_{l,m,f} \tag{3.1}$$

unter den Nebenbedingungen

$$\sum_{m \in A_{l,f}} x_{l,m,f} = 1 \qquad \forall l \in \mathcal{L} \tag{3.2}$$

$$\sum_{k \in B_{l,f}} x_{k,l,f} - \sum_{m \in A_{l,f}} x_{l,m,f} = 0 \qquad \forall l \in \mathcal{L} \tag{3.3}$$

$$\sum_{l \in \mathcal{L}} \sum_{m \in A_{l,f}} \Delta_{l,m,f} x_{l,m,f} \leq N_f \tag{3.4}$$

$$x_{l,m,f} \in \{0,1\} \qquad \forall l \in \mathcal{L}, m \in A_{l,f} \tag{3.5}$$

Einsetzen von (3.2) in (3.1) liefert

$$\text{Maximiere} \sum_{l \in \mathcal{L}} \sum_{m \in A_{l,f}} p_{l,f} x_{l,m,f} = \sum_{l \in \mathcal{L}} p_{l,f} \sum_{m \in A_{l,f}} x_{l,m,f} = \sum_{l \in \mathcal{L}} p_{l,f}$$

und zeigt, dass alle zulässigen Lösungen den selben Gewinn erwirtschaften und die Zielfunktion ignoriert werden kann.

Einsetzen von (3.2) in (3.3) und Berücksichtigen von Gleichung (3.4) als Zielfunktion liefert das Modell:

$$\text{Minimiere} \sum_{l \in \mathcal{L}} \sum_{m \in A_{l,f}} \Delta_{l,m,f} x_{l,m,f} \tag{3.6}$$

unter den Nebenbedingungen

$$\sum_{m \in A_{l,f}} x_{l,m,f} = 1 \qquad \forall l \in \mathcal{L} \tag{3.7}$$

$$\sum_{k \in B_{l,f}} x_{k,l,f} = 1 \qquad \forall l \in \mathcal{L} \tag{3.8}$$

$$x_{l,m,f} \in \{0,1\} \qquad \forall l \in \mathcal{L}, m \in A_{l,f} \tag{3.9}$$

Offensichtlich existiert nun genau dann eine zulässige Zuweisung für das Ursprungsproblem, wenn das transformierte Modell eine Lösung mit Zielfunktionswert kleiner gleich N_f besitzt. Bei dem transformierten Modell handelt es sich aber um ein normales *Assignment Problem*[2] mit $2|\mathcal{L}|$ Knoten und $O(|\mathcal{L}|^2)$ Kanten, dass sich in Zeit $O(|\mathcal{L}|^3)$ lösen lässt [Kuhn, 1955]. □

Satz 3.15. *Die zyklische Klasse FAP*$(Opt, Cy, 1, Ar, Cdt, Cdp, Big)$ *ist NP-vollständig.*

Beweis. Wir reduzieren das ganzzahlige Rucksackproblem

$$\text{Maximiere} \sum_{i \in [n]} v_i x_i \tag{3.10}$$

[2] oder auch *bipartites gewichtetes Matching-Problem*

unter den Nebenbedingungen

$$\sum_{i\in[n]} w_i x_i \leq W \tag{3.11}$$

$$x_i \in \{0,1\} \qquad \forall i \in [n] \tag{3.12}$$

auf die Klasse FAP$(Opt, Cy, 1, Ar, Cdt, Cdp, Big)$. Das ganzzahlige Rucksackproblem gehört zu den klassischen NP-vollständigen Optimierungsproblemen [Karp, 1972].

Die Flottenzuweisungsinstanz besteht aus einer Flotte f mit $W + 2n$ Flugzeugen und $4n$ Legs. Die Planungsperiode erstreckt sich über drei Zeiteinheiten ($\mathcal{T} = 3$). Zu jedem Gut $i \in [n]$ des Rucksacks gehören die Legs l_i^1, l_i^2, l_i^3 und l_i^4. Die Legs verkehren zwischen den $n + 1$ vielen Flughäfen $[n + 1]$, wobei für Gut $i \in [n]$ gilt:

$$\begin{aligned} s^d_{l_i^1} = s^d_{l_i^2} = s^a_{l_i^3} = s^a_{l_i^4} &= n \\ s^a_{l_i^1} = s^a_{l_i^2} = s^d_{l_i^3} = s^d_{l_i^4} &= i \\ t^d_{l_i^1} = t^d_{l_i^2} &= 0 \\ t^d_{l_i^3} = t^d_{l_i^4} &= 1 \\ b_{l_i^1} = b_{l_i^2} = b_{l_i^3} = b_{l_i^4} &= 1 \\ g_{l_i^1, l_i^3, f} &= w_i \cdot \mathcal{T} \\ p_{l_i^1, l_i^3, f} &= v_i \end{aligned}$$

Alle übrigen Gewinne und Mindestbodenzeiten sind Null.

Jede zulässige Flottenzuweisung muss zum Periodenbeginn (mindestens) $2n$ Flugzeuge auf Flughafen n zur Verfügung stellen, da als erstes $2n$ Legs zum Zeitpunkt Null von dort starten. Die Legs selber verbrauchen keine Flugzeuge, da sie nicht über das Periodenende hinaus operieren. Für Flughafen i gibt es nur zwei Möglichkeiten, wie Flugzeuge die dort startenden und landenden Legs bedienen können:

- Ein Flugzeug fliegt nacheinander die Legs l_i^1 und l_i^3, ein anderes nacheinander die Legs l_i^2 und l_i^4. Das bringt einen Gewinn von v_i, verbraucht aber $\lfloor \frac{w_i \cdot \mathcal{T}}{\mathcal{T}} \rfloor = w_i$ Flugzeuge.
- Ein Flugzeug fliegt nacheinander die Legs l_i^1 und l_i^4, ein anderes nacheinander die Legs l_i^2 und l_i^3. Das bringt keine Gewinn, verbraucht aber auch keine Flugzeuge auf Flughafen i.

Mit Flughafen i lässt sich also direkt die Entscheidung modellieren, ob Gut i in den Rucksack aufgenommen werden soll oder nicht.

Damit existiert genau dann eine Lösung des Rucksackproblems mit Wert v^*, wenn das konstruierte Flottenzuweisungsproblem eine Lösung mit Gewinn v^* besitzt. □

Satz 3.16. *Instanzen der azyklischen Klasse FAP$(Opt, Ac, 1, Ar, Cdt)$ lassen sich in Zeit $O(|\mathcal{L}|^{2.5})$ lösen.*

Beweis. Da der erzielbare Gewinn jeder zulässigen Lösung wie im Beweis zu Satz 3.12 gleich ist, müssen wir wieder nur testen, ob wir mit der vorgegebenen Flottengröße auskommen.

Wie wir in Abschnitt 2.2.3 dargelegt haben, lässt sich die Anzahl benötigter Flugzeuge einer gegebenen Flottenzuweisung im azyklischen Fall dadurch ermitteln, dass auf den einzelnen Flughafen gezählt wird, wie viele Flugzeuge dort jeweils ihren Dienst beenden. Dafür ist es nötig, neben der eigentlichen Flottenzuweisung zu jedem Leg l sein etwaiges Nachfolgeleg m zu kennen, das heißt das Leg, mit dem das Flugzeug, das Leg l geflogen hat, seine Reise fortsetzt. Legs ohne Nachfolgeleg sind gleichbedeutend mit einem Flugzeug, das auf dem entsprechenden Zielflughafen seinen Einsatz beendet. Entsprechend zeigt ein Leg ohne Vorgängerleg an, dass ein Flugzeug auf dem entsprechenden Startflughafen seinen Dienst aufnimmt.

Das Problem, zu einer gegebenen Flottenzuweisung z eine flugzeugminimale Verknüpfung der Legs zu finden, lässt sich als bipartites Matching-Problem formulieren. Jedes Leg l wird dabei durch zwei Knoten l^a und l^d repräsentiert, die für die Landung bzw. den Start des Legs stehen. Zwischen einem Landeknoten l^a und einem Startknoten m^d existiert genau dann eine Kante, wenn die betroffenen Legs nacheinander von einem Flugzeug bedient werden können, das heißt, wenn $z(l) = z(m)$, $s^a_l = s^d_m$ und $t^a_{l,m,f} \leq t^d_{m,f}$ gilt. Der so konstruierte Graph ist offensichtlich bipartit und besteht aus $|\mathcal{L}|$ Knoten je Partition und $O(|\mathcal{L}|^2)$ Kanten. Ein maximales Matching für diesen Graphen lässt sich somit in Zeit $O(|\mathcal{L}|^{2.5})$ bestimmen [Even and Tarjan, 1975], [Hopcroft and Karp, 1973] und die Anzahl Landeknoten, die von keiner Kante überdeckt werden, entspricht der Anzahl benötigter Flugzeuge.

Für den Fall, dass die Verteilungen der Flugzeuge auf den Flughäfen zu Periodenbeginn und/oder zum Periodenende vorgegeben sind, wertet man die Anzahl nicht überdeckter Lande- bzw. Start-Knoten flughafenweise aus und vergleicht diese mit den vorgegebenen Verteilungen. □

Satz 3.17. *Instanzen der azyklischen Klasse FAP*$(Opt, Ac, 1, Ar, Cdt, Cdp)$ *lassen sich in Zeit* $O(|\mathcal{L}|^3)$ *lösen.*

Beweis. Ähnlich wie im Beweis zu Satz 3.16 transformieren wir eine Flottenzuweisungsinstanz in ein Matching-Problem, diesmal allerdings ein *gewichtetes* bipartites Matching-Problem (Assignment Problem). Die Grundkonstruktion ist die selbe wie im Beweis zu Satz 3.16. Eine Kante zwischen den Knoten l^a und m^d bekommt dabei das Gewicht $p_{l,m,f}$ zugewiesen. Damit wird durch das Matching der verbindungsabhängige Teil des Gesamtgewinns maximiert.[3]

Um nun noch die Anzahl verfügbarer Flugzeuge einzuhalten und da beim Assignment Problem nur vollständige Matchings eine Lösung darstellen, werden N_f zusätzliche Lande- und N_f zusätzliche Startknoten dem Netzwerk hinzugefügt. Jeder der zusätzlichen Landeknoten ist mit allen Startknoten (auch den zusätzlichen) über eine Kante verbunden. Entsprechend sind die zusätzlichen Startknoten mit allen Landeknoten verbunden. Die zusätzlichen Lande- bzw. Startknoten repräsentieren explizit den Dienstbeginn bzw. -ende eines Flugzeugs.

Für große N_f ist obige Konstruktion nicht in polynomieller Zeit durchführbar. Allerdings können wir ohne Beschränkung der Allgemeinheit N_f durch $|\mathcal{L}|$ begrenzen, da insgesamt

[3] Der Leg-Flotten-abhängige Teil des Gewinns ist für jede Lösung wieder gleich.

höchstens $|\mathcal{L}|$ Flugzeuge überhaupt einen Flug durchführen können. Somit besteht obiger Graph aus höchstens $2|\mathcal{L}|$ Knoten je Partition und $O(|\mathcal{L}|^2)$ Kanten. Kanten zwischen den zusätzlichen Knoten sind notwendig, damit ein Teil der Flugzeuge komplett untätig bleiben kann.

Für den Fall, dass die Verteilungen der Flugzeuge auf den Flughäfen zu Periodenbeginn und/oder zum Periodenende vorgegeben sind, fügt man (im Falle des Periodenbeginns) anstatt der N_f zusätzlichen Landeknoten $N^b_{s,f}$ zusätzliche Landeknoten für jeden Flughafen s ein. Die zusätzlichen Landeknoten eines Flughafens s werden dabei nur mit Startknoten verbunden, deren Leg von s startet. Analog verfährt man für das Periodenende.

Es ist leicht zu zeigen, dass jedes vollständige Matching auf diesem Graphen mit einer zulässigen Flottenzuweisung korrespondiert und dass ein gewichtmaximales vollständiges Matching eine gewinnmaximale Flottenzuweisung repräsentiert. Ein gewichtmaximales vollständiges Matching kann in Zeit $O(|\mathcal{L}|^3)$ bestimmt werden [Kuhn, 1955]. □

3.4 Der Zwei-Flotten-Fall

In diesem Abschnitt präsentieren wir ein neues Vollständigkeitsergebnis: Zu entscheiden, ob ein Flottenzuweisungsproblem, egal ob zyklisch oder azyklisch, eine zulässige Lösung besitzt, ist bereits für zwei Flotten streng NP-vollständig. Bisher war dies nur für drei Flotten bekannt. Allerdings müssen wir für unsere Reduktion erlauben, dass die eingesetzten Flotten unterschiedliche Geschwindigkeiten besitzen.

Satz 3.18. *Die Klassen FAP$(Feas, t, f, Or)$ mit $t \in \{Cy, Ac\}$ und $f \geq 2$ sind streng NP-vollständig.*

Wir beweisen diesen Satz, indem wir SAT auf die FAP$(Feas, 2, f, Or)$-Klasse reduzieren. SAT war das erste Problem, für das gezeigt wurde, dass es NP-vollständig ist [Cook, 1971]. Dazu bilden wir die Arbeitsweise einfacher Boolscher Schaltkreise mit Hilfe spezieller Flughafenkonstrukte nach, die wir vor dem eigentlichen Beweis zunächst vorstellen. Legs zwischen Flughafenkonstrukten übernehmen die Rolle von elektrischen Leitungen. Alle Flughafenkonstrukte sind bezüglich ihrer Starts- und Landungen balanciert, so dass sie sowohl für zyklische als auch azyklische Flottenzuweisungsprobleme eingesetzt werden können.

Für die gesamte Reduktion nehmen wir an:

- Es kommen zwei Flotten zum Einsatz. Eine repräsentiert den Wahrheitswert *Wahr* und wird im Folgenden mit T bezeichnet. Die andere repräsentiert den Wahrheitswert *Falsch* und wird mit F bezeichnet.
- *Flotte T ist die schnellere der beiden Flotten und kann jedes Leg zwei Zeiteinheiten schneller fliegen als Flotte F.* Wir geben daher im Folgenden nur für Flotte T die Blockzeiten bzw. Ankunftszeiten von Legs an. Die entsprechenden Zeiten für Flotte F ergeben sich direkt daraus.
- Die Startzeiten aller Legs sind unabhängig von der verwendeten Flotte.

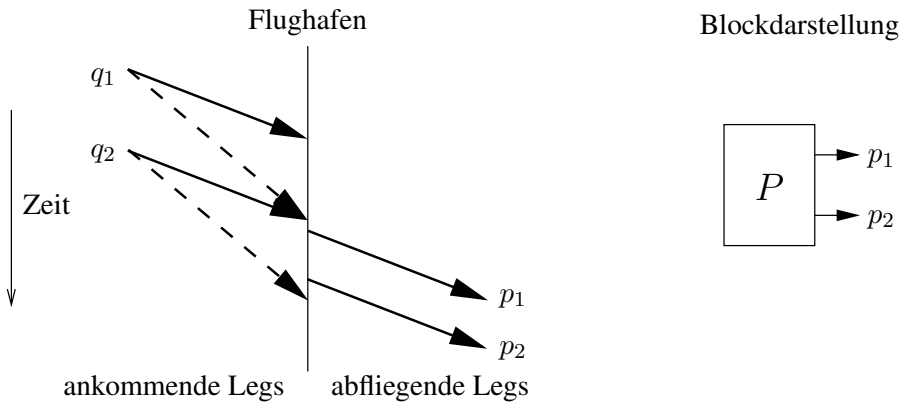


Abbildung 3.1: Flughafenkonstrukt Paar $P(t)$

- Alle Mindestbodenzeiten sind Null.

Die nun folgenden Flughafenkonstrukte sind nur Ausschnitte eines vollständigen Flugplans. Sie definieren die Starts und Landungen von Legs auf einem oder mehreren Flughäfen. Dabei werden die beteiligten Legs häufig nicht vollständig spezifiziert, das heißt, dass zum Beispiel für ein von einem Flughafenkonstrukt startenden Leg zwar die Startzeit definiert wird, nicht aber die Blockzeit/Ankunftszeit und der Zielflughafen. Erst die spätere Verknüpfung der Flughafenkonstrukte miteinander spezifiziert die Legs vollständig und definiert einen vollständigen Flugplan. Die Flughafenkonstrukte sind parametrisiert, um die exakten Start- und Landezeiten der beteiligten Legs variieren zu können.

Wir werden im Anschluss an die Definition eines jeden Flughafenkonstrukts seine zentralen Eigenschaften im Hinblick auf die folgende Reduktion beweisen. Die korrekte Funktionsweise eines Flughafenkonstrukts ist dabei nur dann gewährleistet, wenn zu Beginn der Planungsperiode keinerlei Flugzeuge auf dem Flughafen/den Flughäfen des Konstrukts verfügbar sind. Diese Voraussetzung wird von der Reduktion sichergestellt.

Definition 3.19 (Paar $P(t)$). *Ein Paar $P(t)$ besteht aus einem Flughafen mit zwei ankommenden Legs q_1 und q_2 und zwei abfliegenden Legs p_1 und p_2. Die Start- und Landezeiten sind dabei wie folgt definiert:*

$$\begin{aligned} t^a_{q_1,T} &= t \\ t^a_{q_2,T} &= t+2 \\ t^d_{p_1} &= t+2 \\ t^d_{p_2} &= t+3 \end{aligned}$$

Man beachte, dass dadurch implizit $t^a_{q_1,F} = t+2$ und $t^a_{q_2,F} = t+4$ sind.

Abbildung 3.1 zeigt auf der linken Seite eine schematische Darstellung eines Paares. Die senkrechte Linie in der Mitte symbolisiert den Flughafen. Die Pfeile auf der rechten Seite stehen für abfliegende Legs, wobei die Pfeilenden den Startzeitpunkt auf dem Flughafen markieren.

Die Zeitachse verläuft dabei von oben nach unten. Die ankommenden Legs auf einen Flughafen werden auf der linken Seite dargestellt. Die Pfeilspitzen markieren hier die Ankunftszeit. Da die Ankunftszeit eines Legs von der eingesetzten Flotte abhängig ist, wird jedes ankommende Leg durch zwei Pfeile mit gemeinsamem Endpunkt dargestellt: Der durchgehende Pfeil markiert die Ankunftszeit, wenn das Leg von Flotte T bedient wird, und der gestrichelte Pfeil die Ankunftszeit mit Flotte F. Der Übersichtlichkeit halber werden Ereignisse, die eigentlich gleichzeitig auf einem Flughafen stattfinden, in einer schematischen Darstellung manchmal entzerrt. Dies hat dann aber keinen Einfluss auf die korrekte Funktionsweise eines Flughafenkonstrukts.

Jedes Flughafenkonstrukt hat eine logische Funktion, die bei der Reduktion einer SAT-Formel zum Einsatz kommt. Zur Beschreibung dieser logischen Funktion sind nicht immer alle Legs eines Flughafenkonstrukts relevant. Daher existiert zu jedem Flughafenkonstrukt eine abstrakte Blockdarstellung, die nur die für die logische Funktion notwendigen Legs *ohne Zeitbezug* darstellt. Die Blockdarstellung eines Paares findet sich in der rechten Hälfte von Abbildung 3.1. Hier zeigt sich, dass die Hauptaufgabe eines Paares die Bereitstellung von zwei „Ausgangsleitungen“ ist. Dabei gilt:

Lemma 3.20. *Wenn für ein Paar $P(t)$ zu Beginn der Planungsperiode keine Flugzeuge auf dem Paar-Flughafen warten, kann $P(t)$ nur genau dann Teil einer zulässigen Flottenzuweisung z sein, wenn $z(q_2) = T$ ist. Wird ferner das Leg q_1 des Paares von Flotte F bedient, müssen in einer zulässigen Flottenzuweisung entweder $z(p_1) = T \wedge z(p_2) = F$ oder $z(p_1) = F \wedge z(p_2) = T$ sein.*

Beweis. Wenn $z(q_2) = F$ wäre, würde Leg q_2 erst zum Zeitpunkt $t+4$ landen. Im azyklischen Fall stünden dann aber für die beiden abfliegenden Legs p_1 und p_2 nur das Flugzeug des Legs q_1 als Vorgänger zur Verfügung und eines der beiden Legs könnte nicht bedient werden. Im zyklischen Fall müsste das Flugzeug von q_2 über das Periodenende hinaus auf seinen nächsten Flug warten, würde also ein wartendes Flugzeug zu Periodenbeginn verursachen. Folglich muss $z(q_2) = T$ sein.

Damit stehen für die Legs p_1 und p_2 je ein Flugzeug der Flotten T und F zur Verfügung und die beiden einzigen zulässigen Zuweisungen an p_1 und p_2 sind $z(p_1) = T \wedge z(p_2) = F$ oder $z(p_1) = F \wedge z(p_2) = T$. □

Die logische Funktion eines Paares ist es damit, zwei Legs mit komplementärer Flottenzuweisung bereitzustellen.

Definition 3.21 (Oder $O_n(t)$). *Ein Oder $O_n(t)$ ($n \in \mathbb{N}$) besteht aus einem Flughafen mit n ankommenden Legs $i_1, \ldots, i_n$ und n abfliegenden Legs $h_1, \ldots, h_n$. Die Start- und Landezeiten sind dabei wie folgt definiert:*

$$
\begin{aligned}
t^a_{i_j,T} &= t && \forall j \in \{1, \ldots, n\} \\
t^d_{h_1} &= t \\
t^d_{h_j} &= t + 2 && \forall j \in \{2, \ldots, n\}
\end{aligned}
$$

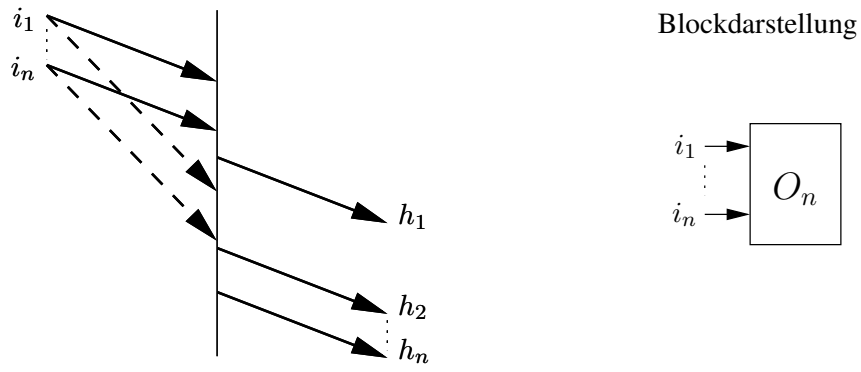


Abbildung 3.2: Flughafenkonstrukt Oder $O_n(t)$

Lemma 3.22. *Wenn für ein Oder $O_n(t)$ zu Beginn der Planungsperiode keine Flugzeuge auf dem Oder-Flughafen warten, kann $O_n(t)$ nur genau dann Teil einer zulässigen Flottenzuweisung z sein, wenn ein Leg i_j ($j \in \{1, \ldots, n\}$) mit $z(i_j) = T$ existiert.*

Beweis. Werden alle ankommenden Legs i_j von Flotte F bedient, landen sie alle nach dem Start von Leg h_1. h_1 könnte also entweder nicht bedient werden oder ein Flugzeug würde zu Periodenbeginn auf dem Oder-Flughafen warten müssen.

Umgekehrt reicht ein Leg i_j mit $z(i_j) = T$ aus, um das abfliegende Leg h_1 zu erreichen. Alle übrigen Legs h_k starten nach der Ankunft der restlichen i_k, unabhängig von deren Flottenzuweisung. □

Abbildung 3.2 zeigt einen schematischen Oder-Flughafen und dessen Blockdarstellung.

Definition 3.23 (Duplikator $D_2(t)$). *Ein Duplikator $D_2(t)$ besteht aus einem Flughafen mit drei ankommenden Legs i, x_1 und x_2 und drei abfliegenden Legs $h_1, \ldots, h_3$. Die Start- und Landezeiten sind dabei wie folgt definiert:*

$$
\begin{aligned}
t^a_{x_1,T} &= t + 4 \\
t^a_{i,T} &= t + 5 \\
t^a_{x_2,T} &= t + 6 \\
t^d_{h_1} &= t + 5 \\
t^d_{h_2} &= t + 6 \\
t^d_{h_3} &= t + 8
\end{aligned}
$$

Zusätzlich enthält der Duplikator noch zwei Paare $P_1(t)$ und $P_2(t)$. Das abfliegende Leg p_2 von Paar $P_j(t)$ ist dabei identisch mit Leg x_j und das abfliegende Leg p_1 von Paar $P_j(t)$ wird im Duplikator o_j genannt ($j \in \{1, 2\}$).

Abbildung 3.3 zeigt ein schematisches Duplikator-Konstrukt und dessen Blockdarstellung. Man beachte, dass die Legs x_j im Duplikator vollständig spezifiziert sind und echt positive Blockzeiten besitzen. Die logische Funktion eines Duplikators ist, die Flottenzuweisung seines

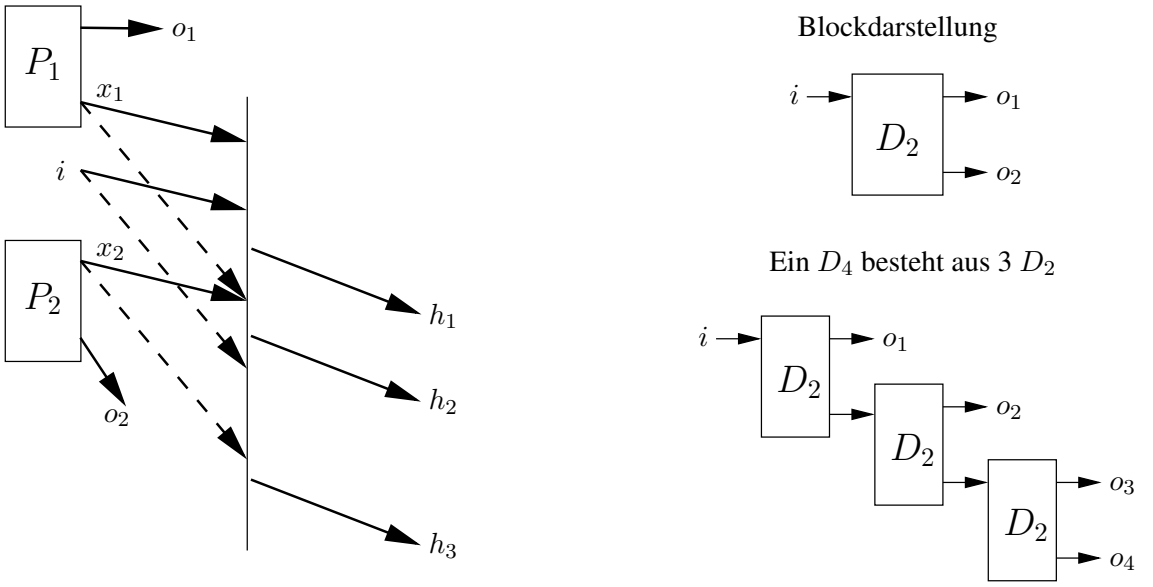


Abbildung 3.3: Flughafenkonstrukt Duplikator $D_n(t)$

Legs i auf die Legs o_1 und o_2 zu übertragen. Dabei darf er bei einer Zuweisung $z(i) = T$ fehlerhaft arbeiten, wichtig ist nur, dass in diesem Fall $z(o_1) = z(o_2) = T$ möglich ist.

Lemma 3.24. *Wenn für einen Duplikator $D_2(t)$ zu Beginn der Planungsperiode keine Flugzeuge auf dem Duplikator-Flughafen warten und für seine Paare die Voraussetzungen von Lemma 3.20 gelten, dann gilt:*

1. *$D_2(t)$ kann nur dann Teil einer zulässigen Zuweisung z mit $z(i) = F$ sein, wenn $z(o_1) = z(o_2) = F$ gilt.*
2. *Aus Sicht des Operators $D_2(t)$ gibt es eine zulässige Zuweisung z mit $z(i) = z(o_1) = z(o_2) = T$.*

Beweis.

1. Wenn $z(i) = F$ ist, landet Leg i nach den Starts von Leg h_1 und h_2. Daher muss $z(x_1) = z(x_2) = T$ gelten. Nach Lemma 3.20 gilt dann aber $z(o_1) = z(o_2) = F$.
2. Die Zuweisung $z(i) = z(h_1) = z(o_1) = z(o_2) = T$ und $z(x_1) = z(x_2) = z(h_2) = z(h_3) = F$ ist aus Sicht des Duplikators zulässig und erfüllt die Voraussetzungen des Lemmas.

□

Für unsere Reduktion benötigen wir Duplikatoren mit mehr als zwei „Ausgangsleitungen“.

Definition 3.25 (Duplikator $D_n(t)$). *Ein Duplikator $D_n(t)$ ($n > 2$) besteht aus $n - 1$ Duplikatoren $D_2(t)$, die wie in Abbildung 3.3 für $D_4(t)$ gezeigt kaskadiert werden.*

Man beachte, dass alle D_2-Duplikatoren in solch einer Kaskadierung zum selben Zeitpunkt t „starten“ und dass die dadurch vollständig spezifizierten Legs zwischen den D_2-Duplikatoren allesamt echt positive Blockzeiten besitzen. Wie man leicht sieht, gilt Lemma 3.24 auch für allgemeine Duplikatoren $D_n(t)$.

Wir kennen nun alle Zutaten, um eine SAT-Instanz in ein Flottenzuweisungsproblem zu transformieren.

Definition 3.26 (Transformation einer SAT-Instanz B in einen Flottenzuweisungsinstanz $FA(B)$). *Sei $B = \bigwedge_{i \in C} \bigvee_{j=1}^{c_i} l_{ij}$ eine Boolsche Formel in konjunktiver Normalform, wobei C die Menge der Klauseln, $c_i \in \mathbb{N}$ die Länge von Klausel $i \in C$ und $l_{i,j}$ das j-te Literal von Klausel $i \in C$ ist. V sei die Menge der Variablen in B, und $|l|$ bezeichne die Häufigkeit von Literal l in B.*

Wir transformieren B wie folgt in eine Flottenzuweisungsinstanz $FA(B)$:

- *Für jede Variable $v \in V$ generieren wir ein Paar $P_v(1)$. Die beiden Ausgänge von $P_v(1)$ werden dabei mit den Literalen v und $\bar{v}$ markiert.*
- *Für jedes Literal l mit $|l| > 1$ generieren wir einen $D_{|l|}(1)$-Duplikator, dessen Eingangsleg i mit dem mit l markierten Ausgangsleg eines Variablen-Paares gleichgesetzt wird. Alle Ausgänge des Duplikators werden mit l markiert.*
- *Für jede Klausel $i \in C$ generieren wir ein Oder $O_{c_i}(4)$ und verknüpfen seine c_i Eingangslegs mit zu den Literalen l_{ij} korrespondierenden, noch nicht vollständig spezifizierten Ausgangslegs von Literal-Duplikatoren bzw. Variablen-Paaren. Die mit einem Literal l markierten Ausgangslegs von Duplikatoren bzw. Paaren reichen nach Konstruktion aus, um die Eingänge aller Klausel-Oders zu verknüpfen.*
- *Nicht alle Legs sind dadurch bis jetzt voll spezifiziert worden. Legs ohne spezifizierten Startflughafen (und Abflugzeit) sind nach Konstruktion ausnahmslos die Eingänge von Paaren. Sei N die Anzahl dieser Paare. Die $2N$ Eingangslegs der Paare starten alle zum Zeitpunkt Null auf einem zusätzlichen Flughafen XYZ.*
- *Alle jetzt noch nicht vollständig spezifizierten Legs (ausnahmslos Ausgangslegs) landen zum Zeitpunkt 9 auf Flughafen XYZ, wenn sie von Flotte T bedient werden. (Die Landezeit für Flotte F ist damit 11.)*
- *Wir stellen insgesamt je N Flugzuge von Flotte T und F zur Verfügung. Wenn wir für die azyklischen Untervarianten $Ac1$ bzw. $Ac2$ die Flugzeugverteilung zum Periodenanfang und/oder -ende spezifizieren müssen, sagen wir, dass sich die Flugzeuge aller Flotten zum Periodenanfang bzw. zum Periodenende auf dem Flughafen XYZ aufhalten sollen.*
- *Die Periodenlänge im zyklischen Fall beträgt $\mathcal{T} = 12$.*

Die Paaranzahl N setzt sich aus den Variablen-Paaren und den Paaren in den Literal-Duplikatoren zusammen. Da ein Duplikator $D_n(t)$ aus $2(n-1)$ Paaren besteht gilt:

$$N = |V| + \sum_{l \text{ Literal}:|l| \geq 2} 2(|l| - 1)$$

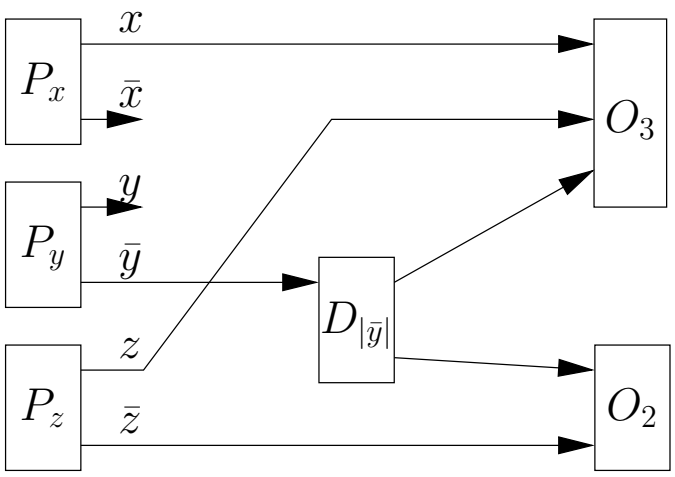


Abbildung 3.4: Die Boolsche Formel $(x \vee \bar{y} \vee z) \wedge (\bar{y} \vee \bar{z})$ transformiert in eine Flottenzuweisungsinstanz

Da alle Flughafen-Konstrukte balanciert sind, muss das auch für den Flughafen XYZ gelten, das heißt, auf XYZ enden auch insgesamt $2N$ Legs.

Man prüft leicht nach, dass durch die Konstruktion in Definition 3.26 eine formal korrekte Flottenzuweisungsinstanz $FA(B)$ erzeugt wird. Insbesondere sind sämtliche (implizit durch die Start- und Landezeiten definierten) Blockzeiten echt positiv.

Abbildung 3.4 zeigt das Ergebnis obiger Transformation für die Boolsche Formel $(x \vee \bar{y} \vee z) \wedge (\bar{y} \vee \bar{z})$ in Blockdarstellung.

Beweis von Satz 3.18. Wir reduzieren SAT auf FAP$(Feas, t, f, Or)$ und verwenden dazu die Transformation aus Definition 3.26. Die Transformation lässt sich offensichtlich in polynomieller Zeit ausführen und der Betrag sämtlicher in der Flottenzuweisungsinstanz auftretenden Zahlen ist polynomiell in der Instanzgröße beschränkt.

Für jede zulässige Zuweisung einer Flottenzuweisungsinstanz $FA(B)$ gilt nach Konstruktion:

- Da zu Periodenbeginn von Flughafen XYZ insgesamt $2N$ Legs starten und uns insgesamt nur $2N$ Flugzeuge zur Verfügung stehen, müssen sich alle Flugzeuge zu Periodenbeginn auf Flughafen XYZ aufhalten.
- Da später auf Flughafen XYZ insgesamt $2N$ Legs enden, befinden sich zum Periodenende alle Flugzeuge wieder auf Flughafen XYZ.
- Ferner kann sich zu Periodenbeginn auf keinem anderen Flughafen ein Flugzeug aufhalten.
- Nach Lemma 3.20 müssen daher die q_2-Legs aller N Paare von Flotte T bedient werden. Da die q_i-Legs aller Paare von Flughafen XYZ starten, bleiben für die q_1-Legs der Paare nur Flugzeuge der Flotte F übrig.

- Daher erfüllt jede zulässige Zuweisung die Voraussetzungen der Lemmata 3.20, 3.22 und 3.24.

- Vorgegebene Verteilungen der Flugzeuge der beiden Flotten (Untervarianten $Ac1$ und $Ac2$) werden implizit erfüllt.

Es verbleibt zu zeigen, dass eine Boolsche Formel B genau dann erfüllbar ist, wenn die Flottenzuweisungsinstanz $FA(B)$ eine zulässige Lösung besitzt.

$\Rightarrow$ Sei $b : V \to \{F, T\}$ eine erfüllende Variablenbelegung der Formel B. Den Eingangslegs aller Paare wird gemäß obiger Beschreibung eine Flotte zugewiesen. Allen mit Literal v markierten Legs in $FA(B)$ wird die Flotte $b(v)$ zugewiesen und allen mit $\bar{v}$ markierten Legs wird die Flotte $\overline{b(v)}$ zugewiesen ($v \in V$). Damit werden den Ausgängen der Variablen-Paare komplementäre Flotten zugewiesen und die Flottenbelegung der i- und o_j-Legs eines jeden Duplikators ist identisch. Da b erfüllend ist, besitzt ferner jedes Oder-Flughafenkonstrukt mindestens ein Eingangsleg mit Flottenzuweisung T. Somit lassen sich nach den Lemmata 3.20, 3.22 und 3.24 allen verbleibenden Legs Flotten derart zuweisen, dass insgesamt eine zulässige Zuweisung entsteht.

$\Leftarrow$ Sei z eine zulässige Zuweisung für $FA(B)$. Daraus konstruieren wir eine Variablenbelegung b für die Formel B, indem wir $b(v)$ die Flotte zuweisen, die das mit Literal v markierte Ausgangsleg des Variablen-Paares $P_v(1)$ zugewiesen bekommen hat. b ist dann eine erfüllende Variablenbelegung für Formel B.

Annahme: b ist nicht erfüllend.

Dann gibt es eine Klausel i in B, deren Literale allesamt F liefern. Da z aber zulässig ist, muss nach Lemma 3.22 eines der Eingangslegs des zu i gehörenden Oder-Flughafenkonstrukts von Flotte T bedient werden. Dieses Leg bezeichnen wir mit L, und es sei mit Literal l markiert. v sei die Variable des Literals l.

L kann zwei verschiedene Arten von Startflughafen besitzen. Der Startflughafen von L kann zum Variablen-Paar $P_v(1)$ gehören. Ansonsten gehört der Starflughafen von L zum Duplikator $D_{|l|}(1)$. Das Eingangsleg i dieses Duplikators muss dann ebenfalls von Flotte T bedient werden, da ansonsten nach Lemma 3.24 alle Ausgänge des Duplikators F sein müssen. Auch i ist mit Literal l markiert und der Startflughafen von i gehört zum Variablen-Paar $P_v(1)$.

Also wird das mit l markierte Ausgangsleg des Variablen-Paares $P_v(1)$ von Flotte T bedient. Ist $l = v$, ist nach Definition $b(v) = T$ und Klausel i von B wäre erfüllt. Ist $l = \bar{v}$, wird nach Lemma 3.20 das Ausgangsleg von $P_v(1)$, das mit v markiert ist, von Flotte F bedient. Also ist $b(v) = F$ und Klausel i von B wäre auch in diesem Fall erfüllt. Dies liefert den gewünschten Widerspruch. □

Folgerung 3.27. *Die Klassen FAP(Opt, t, f, Or) mit $t \in \{Cy, Ac\}$ und $f \geq 2$ sind streng NP-vollständig und nicht in polynomieller Zeit approximierbar, falls P $\neq$ NP gilt.*

Beweis. Die Argumentation verläuft wie im Beweis zu Folgerung 3.6. □

Die Reduktion in diesem Kapitel funktioniert nur dann, wenn die Flugzeugflotten verschiedene Geschwindigkeiten besitzen. *Erlaubt man verbindungsabhängige Gewinne*, kann man aber ein vergleichbares Ergebnis auch für gleich schnelle Flotten zeigen:

Folgerung 3.28. *Die Klassen FAP*(Opt, t, f, Eq, Cdp) *mit* $t \in \{Cy, Ac\}$ *und* $f \geq 2$ *sind streng NP-vollständig.*

Beweis. Die Reduktion verläuft wie im Beweis zu Satz 3.18. Flotte F ist nun aber genauso schnell wie Flotte T. Dadurch mögliche, zusätzliche Verbindungen für Flotte F werden mit verbindungsabhängigen Gewinnen von -1 bestraft. Alle übrigen Gewinne sind Null.

Wie man leicht sieht, gilt dann, dass eine Boolsche Formel B genau dann erfüllbar ist, wenn es eine Flottenzuweisung mit einem Gewinn von (mindestens) Null gibt. □

3.5 Weitere azyklischen Ergebnisse

Hier präsentieren wir Komplexitätsaussagen für azyklische Flottenzuweisungsprobleme, wobei wir an die Flotten die gleichen restriktiven Anforderungen wie [Gu et al., 1994] stellen.

Satz 3.29. *Instanzen der Klasse FAP*$(Feas, Ac0, f, Eq)$ *mit* $f \geq 1$ *lassen sich in Zeit* $O(|\mathcal{L}| \log |\mathcal{L}| + |\mathcal{F}|)$ *lösen.*

Beweis. Dieses Problem lässt sich für $f > 1$ direkt auf FAP$(Feas, Ac0, 1, Eq)$ reduzieren. Da sich die einer Flotte zugewiesenen Legs im azyklischen Fall in Flugzeugumläufe aufteilen lassen, die jeweils genau ein Flugzeug benötigen, und da die Flugzeuge aller Flotten in der Klasse FAP$(Feas, Ac0, f, Eq)$ aus operationeller Sicht identisch sind, lassen sich die Umläufe einer Ein-Flotten-Lösung beliebig auf mehrere Flotten aufteilen, solange insgesamt genügend Flugzeuge zur Verfügung stehen.

Das Bestimmen der Gesamtflugzeuganzahl über alle Flotten benötigt Zeit $O(|\mathcal{F}|)$ und der Ein-Flotten-Fall ist in Zeit $O(|\mathcal{L}| \log |\mathcal{L}|)$ entscheidbar (siehe Satz 3.12). □

Satz 3.30. *Instanzen der Klasse FAP*$(Feas, Ac1, f, Eq)$ *mit* $f \geq 1$ *lassen sich in Zeit* $O(|\mathcal{L}| \log |\mathcal{L}| + |\mathcal{F}| \cdot |\mathcal{S}|)$ *lösen.*

Beweis. Auch dieses Problem lässt sich mit derselben Begründung wie im Beweis zu Satz 3.29 für $f > 1$ direkt auf FAP$(Feas, Ac1, 1, Eq)$ reduzieren. Das Bestimmen der Gesamtflugzeuganzahl pro Flughafen über alle Flotten benötigt Zeit $O(|\mathcal{F}| \cdot |\mathcal{S}|)$. □

Die Argumentation der vorherigen beiden Beweise lässt sich nicht auf die Klasse FAP$(Feas, Ac2, f, Eq)$ übertragen, da man hierfür bei der Generierung eines Flugzeugumlaufs explizit den Start- *und* Zielflughafen vorgeben können müsste. Dies ist sehr wahrscheinlich schwierig, denn es gilt:

Satz 3.31. *Die Klasse FAP*$(Feas, Ac2, f, Eq)$ *mit* $f \geq 3$ *ist streng NP-vollständig.*

Der Beweis beruht auf einer Reduktion von dem gerichteten ganzzahligen Zwei-Güter-Flussproblem mit Kantenkapazitäten von 1. Das Problem lässt sich wie folgt definieren:

Definition 3.32 (Das gerichtete ganzzahlige Zwei-Güter-Flussproblem mit Kantenkapzitäten von 1).

Gegeben: Ein gerichteter Graph $G = (V, E)$, zwei Knotenpaare $(s_1, t_1), (s_2, t_2) \in V^2$ und zwei Transportbedarfe $R_1, R_2 \in \mathbb{N}$.

Existieren in G R_1 Wege von s_1 nach t_1 und R_2 Wege von s_2 nach t_2, die kantendisjunkt sind?

In [Even et al., 1976] wurde gezeigt:

Satz 3.33. *Das gerichtete ganzzahlige Zwei-Güter-Flussproblem mit Kantenkapazitäten von 1 ist streng NP-vollständig. Dies gilt sogar, wenn der Graph azyklisch ist.*

Beweis von Satz 3.31. Das Zwei-Güter-Flussproblem sei durch den gerichteten azyklischen Graphen $G = (V, E)$, die Knotenpaare $(s_1, t_1), (s_2, t_2) \in V^2$ und $R_1, R_2 \in \mathbb{N}$ gegeben. Die Knotenmenge $V = \{v_1, \ldots, v_n\}$ sei topologisch sortiert. $in(v_i)$ sei der Eingangsgrad von Knoten $v_i \in V$ und $out(v_i)$ sein Ausgangsgrad.

Wir konstruieren aus G wie folgt einen Flugplan:

- Jeder Knoten des Graphen $v_i \in V$ wird zu einem Flughafen.
- Jede Kante $(v_i, v_j) \in E$ wird zu einem Leg mit Startflughafen v_i und Zielflughafen v_j.
- Alle Legs, die von Flughafen v_i starten, besitzen Startzeit i.
- Die Blockzeit aller Legs ist 1.
- Alle Mindestbodenzeiten sind Null.

Wegen der topologischen Sortierung der Knoten (und der Blockzeiten von 1) ist sichergestellt, dass alle Legs, die auf einem Flughafen v_i landen, eine Ankunftszeit kleiner gleich i besitzen und damit alle von v_i startenden Legs erreichen können. Also korrespondiert jeder Weg von s nach t in G eineindeutig mit einem Flugzeugumlauf im Flugplan, der auf Flughafen s beginnt, auf Flughafen t endet und der die den Kanten des Wegs entsprechenden Legs bedient.

Wir benötigen drei Flotten f_1, f_2 und f_3, die wie folgt auf die Flughäfen verteilt werden:

$$
\begin{aligned}
N^b_{s_1,f_1} &= R_1 \\
N^e_{t_1,f_1} &= R_1 \\
N^b_{s_2,f_2} &= R_2 \\
N^e_{t_2,f_2} &= R_2 \\
N^b_{v,f_3} &= \max\{-d(v), 0\} && \forall v \in V \\
N^e_{v,f_3} &= \max\{d(v), 0\} && \forall v \in V
\end{aligned}
$$

Alle nicht spezifizierten $N^b_{s,f}$- und $N^e_{s,f}$-Werte sind Null und $d(v)$ ist wie folgt definiert:

$$d(v) = in(v) - out(v) + N^b_{v,f_1} - N^e_{v,f_1} + N^b_{v,f_2} - N^e_{v,f_2}$$

Die Flotte f_3 ist notwendig, da eine zulässige Flottenzuweisung *allen* Legs eine Flotte zuweisen muss. Flotte f_3 dient ausschließlich dazu, Legs, die weder von f_1 noch von f_2 benutzt werden, „aufzubrauchen".

Offensichtlich lässt sich eine Zwei-Güter-Flussprobleminstanz in polynomieller Zeit in obiges Flottenzuweisungsproblem transformieren, und der Betrag der dabei auftretenden Zahlen ist polynomiell durch die Eingabegröße beschränkt, solange dies auch für R_1 und R_2 gilt.

Es verbleibt noch zu zeigen, dass es genau dann die geforderten kantendisjunkten Wege in G gibt, wenn das konstruierte Flottenzuweisungsproblem eine zulässige Lösung besitzt.

$\Leftarrow$. In einer zulässigen Lösung des Flottenzuweisungsproblems gibt es R_1 Flugzeugumläufe (von Flotte f_1), die auf Flughafen s_1 beginnen und auf Flughafen t_1 enden, da nur auf Flughafen s_1 zu Beginn Flugzeuge der Flotte f_1 verfügbar sind und am Periodenende sich alle diese Flugzeuge auf Flughafen t_1 aufhalten. Entsprechend gibt es R_2 Umläufe (von Flotte f_2), die auf Flughafen s_2 beginnen und auf Flughafen t_2 enden. Da Legs in einer zulässigen Flottenzuweisung nicht mehr als einmal von einem Flugzeug bedient werden dürfen, entsprechen diese Umläufe kantendisjunkten Wegen im Ursprungsgraphen G.

$\Rightarrow$. Wir konstruieren zu einem kantendisjunkten Wegesystem für G wie folgt eine Lösung des Flottenzuweisungsproblems:

- Die Kanten, die zu Wegen zwischen s_1 und t_1 gehören, werden von Flotte f_1 bedient.
- Die Kanten, die zu Wegen zwischen s_2 und t_2 gehören, werden von Flotte f_2 bedient.
- Alle Kanten, denen dann noch keine Flotte zugewiesen wurde, werden von Flotte f_3 bedient.

Damit wird jeder Kante genau eine Flotte zugewiesen. Wegen der Korrespondenz von Wegen in G und Flugzeugumläufen im Flugplan ist diese Zuweisung für die Flotten f_1 und f_2 zulässig, das heißt, sie hält die vorgegebene Flugzeugverteilung für die Flotten f_1 und f_2 ein.

Für einen beliebigen Flughafen v sei nun a_i bzw. d_i die Anzahl an Legs, die auf Flughafen v landen bzw. starten und von Flotte f_i bedient werden ($i \in \{1, 2, 3\}$). Die Flottenzuweisung ist nun auch bezüglich Flotte f_3 zulässig, wenn gilt:

- Falls $a_3 > d_3$, beenden $a_3 - d_3$ Flugzeuge von Flotte f_3 ihren Dienst auf Flughafen v. d_3 ankommende Flugzeuge von Flotte f_3 bedienen als nächstes die d_3 abfliegenden Legs. Dies ist gleichbedeutend damit, dass $N^b_{v,f_3} = 0$ und $N^e_{v,f_3} = a_3 - d_3$ gilt.
- Falls $a_3 = d_3$, bedienen alle ankommenden Flugzeuge von Flotte f_3 als nächstes die d_3 abfliegenden Legs, oder kürzer $N^b_{v,f_3} = 0$ und $N^e_{v,f_3} = 0$.
- Falls $a_3 < d_3$, muss entsprechend $N^b_{v,f_3} = d_3 - a_3$ und $N^e_{v,f_3} = 0$ gelten.

Zusammenfassend ist damit die Zuweisung bezüglich Flotte f_3 dann zulässig, wenn gilt:

$$a_3 - d_3 = N^e_{v,f_3} - N^b_{v,f_3} = \max\{d(v), 0\} - \max\{-d(v), 0\} = d(v)$$

Da unsere Flottenzuweisung für die Flotten f_1 und f_2 zulässig ist, gilt für $i \in \{1, 2\}$:

$$\begin{aligned} & a_i + N^b_{v,f_i} = d_i + N^e_{v,f_i} \\ \iff \quad & d_i - a_i = N^b_{v,f_i} - N^e_{v,f_i} \end{aligned}$$

Daraus folgt nun die Zulässigkeit der Flottenzuweisung auch für Flotte f_3:

$$\begin{aligned} a_3 - d_3 &= (in(v) - a_1 - a_2) - (out(v) - d_1 - d_2) \\ &= in(v) - out(v) + d_1 - a_1 + d_2 - a_2 \\ &= in(v) - out(v) + N^b_{v,f_1} - N^e_{v,f_1} + N^b_{v,f_2} - N^e_{v,f_2} \\ &= d(v) \end{aligned}$$

□

Satz 3.34. *Die Klasse FAP(Opt, Ac, f, Eq) mit $f \geq 3$ ist streng NP-vollständig. Dies gilt bereits für Gewinne aus $\{0, 1\}$.*

Beweis. Für die azyklische Untervariante $Ac2$ folgt das Ergebnis direkt aus Satz 3.31. Für die Varianten $Ac0$ und $Ac1$ erweitern wir die Konstruktion aus dem Beweis von Satz 3.31. Wir können dabei ohne Beschränkung der Allgemeinheit annehmen, dass $R_1 \leq |E|$ und $R_2 \leq |E|$ gilt, da ansonsten klar ist, dass nicht ausreichend viele kantendisjunkte Wege existieren können.[4]

Wir erweitern den Flugplan aus dem Beweis von Satz 3.31 wie folgt:

- es gibt zwei neue Flughäfen s^* und t^*.
- Von s^* starten R_1 Legs zu Flughafen s_1. Die Legs starten zum Zeitpunkt Null. Wenn die Legs von Flotte f_1 bedient werden, erwirtschaften sie einen Gewinn von 1.
- Von s^* starten R_2 Legs zu Flughafen s_2. Die Legs starten zum Zeitpunkt Null. Wenn die Legs von Flotte f_2 bedient werden, erwirtschaften sie einen Gewinn von 1.
- Von Flughafen t_1 fliegen R_1 Legs zum Flughafen t^*. Die Legs starten zum Zeitpunkt n. Wenn die Legs von Flotte f_1 bedient werden, erwirtschaften sie einen Gewinn von 1.
- Von Flughafen t_2 fliegen R_2 Legs zum Flughafen t^*. Die Legs starten zum Zeitpunkt n. Wenn die Legs von Flotte f_2 bedient werden, erwirtschaften sie einen Gewinn von 1.
- Alle übrigen Gewinne sind Null.

[4] Falls $s_i = t_i$ ist, kann es trotzt $R_i > |E|$ eine zulässige Lösung geben. Allerdings haben wir es dann mit einem in polynomieller Zeit entscheidbaren Ein-Güter-Fluss-Problem zu tun.

- Wir stellen R_1 Flugzeuge von Flotte f_1, R_2 Flugzeuge von Flotte f_2 und $|E|$ Flugzeuge von Flotte f_3 zur Verfügung.

Damit gilt weiterhin, das jedes ankommende Leg auf einem Flughafen jedes dort startende Leg erreichen kann. Der maximal erzielbare Gewinn ist offensichtlich $2R_1 + 2R_2$ und wird genau dann erreicht, wenn die neuen Legs von s^* nach s_i bzw. von t_i nach t^* von Flotte f_i bedient werden ($i \in \{1, 2\}$). Dafür müssen alle Flugzeuge der Flotten f_1 und f_2 ihren Dienst auf Flughafen s^* aufnehmen und auf Flughafen t^* beenden. Für eine Lösung mit Gewinn $2R_1 + 2R_2$ erzwingen wir so implizit, dass die Flugzeugumläufe von f_1 und f_2 den gesuchten kantendisjunkten Wegen in G entsprechen. Die verfügbaren Flugzeuge von Flotte f_3 reichen aus, um jedes nicht von f_1 oder f_2 benötigte Leg zu bedienen.

Analog zum Beweis von Satz 3.31 kann man dann zeigen, dass genau dann die geforderten kantendisjunkten Wege in G existieren, wenn es eine zulässige Flottenzuweisung mit einem Gewinn von (mindestens) $2R_1 + 2R_2$ gibt. □

Satz 3.35. *Instanzen der Klasse FAP$(Inf, Ac0, f, Ar, Cdt)$ mit $f \geq 1$ lassen sich in Zeit $O(|\mathcal{L}| \cdot |\mathcal{F}|)$ lösen.*

Beweis. Da uns unbeschränkt viele Flugzeuge von jeder Flotte zur Verfügung stehen, können wir jedes Leg von einem eigenen Flugzeug fliegen lassen. In Zeit $O(|\mathcal{L}| \cdot |\mathcal{F}|)$ kann man für jedes Leg die profitabelste der für das Leg zugelassenen Flotten ermitteln. Die Existenz von verbindungsabhängigen Mindestbodenzeiten spielt folglich keine Rolle. □

3.6 Zusammenfassung

Die wichtigsten Ergebnisse dieses Kapitels sind in den Tabellen 3.1 bis 3.4 zusammengefasst. „NP-v." steht dabei für „NP-vollständig" und „s. NP-v." für „streng NP-vollständig". Für Einträge mit einem „?" ist unbekannt, ob sich die Probleme in polynomieller Zeit lösen lassen. Es ist aber nach Satz 3.3 klar, dass alle Probleme zur Klasse NP gehören.

Für FAP$(Opt, *)$-Problemklassen, deren FAP$(Feas, *)$-Varianten bereits NP-vollständig sind, gilt, dass sie nicht in polynomieller Zeit approximierbar sind, falls P $\neq$ NP ist (Satz 3.6 und 3.27). Dies ist für fast alle Problemklassen mit mehr als zwei Flotten der Fall.

Die bekannten Ergebnisse aus Abschnitt 3.2 finden sich allesamt in Tabelle 3.2. Der Beitrag dieser Arbeit lässt sich in drei Bereiche untergliedern:

- Neue Ergebnisse zur NP-Vollständigkeit für Flottenzuweisungsprobleme mit zwei Flotten
- Vergleich der bekannten Ergebnisse für das zyklische Flottenzuweisungsproblem mit entsprechenden azyklischen Varianten
- Auswirkung von Modellerweiterungen wie verbindungsabhängigen Mindestbodenzeiten und Gewinnen

e	$t = Cy$	$t = Ac$
$[Big]$	$O(\|\mathcal{L}\| \log \|\mathcal{L}\|)$	$O(\|\mathcal{L}\| \log \|\mathcal{L}\|)$
$Cdt[, Big]$	$O(\|\mathcal{L}\|^3)$	$O(\|\mathcal{L}\|^{2.5})$
Cdt, Cdp	? NP-v. für $e = Big$	$O(\|\mathcal{L}\|^3)$

Tabelle 3.1: Algorithmische Komplexität des Flottenzuweisungsproblems mit einer Flotte FAP$(Opt, t, 1, Ar, e)$

p	$f = 1$	$f = 2$	$f \geq 3$
$Feas$	$O(\|\mathcal{L}\| \log \|\mathcal{L}\|)$	? NP-v. für $e = Big$	s. NP-v.
Inf	$O(\|\mathcal{L}\| \log \|\mathcal{L}\|)$	P (min cost flow)	s. NP-v.
Opt	$O(\|\mathcal{L}\| \log \|\mathcal{L}\|)$	? NP-v. für $e = Big$ s. NP-v. für $e = Cdp$	s. NP-v.

Tabelle 3.2: Algorithmische Komplexität des zyklischen Flottenzuweisungsproblems für gleich schnelle Flotten FAP(p, Cy, f, Eq)

p	$t = Ac0, Ac1$			$t = Ac2$		
	$f = 1$	$f = 2$	$f \geq 3$	$f = 1$	$f = 2$	$f \geq 3$
$Feas$	$O(\|\mathcal{L}\| \log \|\mathcal{L}\| + \|\mathcal{L}\| \cdot \|\mathcal{F}\|)$			$O(\|\mathcal{L}\| \log \|\mathcal{L}\|)$	?	s. NP-v.
Inf	$O(\|\mathcal{L}\| \cdot \|\mathcal{F}\|)$			nicht definiert		
Opt	$O(\|\mathcal{L}\| \log \|\mathcal{L}\|)$	? s. NP-v. für $e = Cdp$	s. NP-v.	$O(\|\mathcal{L}\| \log \|\mathcal{L}\|)$	? s. NP-v. für $e = Cdp$	s. NP-v.

Tabelle 3.3: Algorithmische Komplexität des azyklischen Flottenzuweisungsproblems für gleich schnelle Flotten FAP(p, t, f, Eq)

p	$t = Cy$		$t = Ac$	
	$f = 2$	$f \geq 3$	$f = 2$	$f \geq 3$
$Feas$	s. NP-v.		s. NP-v.	
Inf	P	s. NP-v.	$O(\|\mathcal{L}\| \cdot \|\mathcal{F}\|)$	
Opt	s. NP-v.		s. NP-v.	

Tabelle 3.4: Algorithmische Komplexität des Flottenzuweisungsproblems für Flotten mit unterschiedlichen Geschwindigkeiten FAP(p, t, f, Or)

Die Reduktion in Abschnitt 3.4 zeigt, dass das Zulässigkeitsproblem bereits für zwei Flotten streng NP-vollständig ist, wenn man Flotten mit unterschiedlichen Geschwindigkeiten erlaubt. Dies gilt sowohl für zyklische als auch azyklische Flottenzuweisungsprobleme. Daraus folgt direkt die strenge NP-Vollständigkeit für Optimierungsprobleme mit zwei gleich schnellen Flotten und verbindungsabhängigen Gewinnen. Damit kann man sagen, dass das Problem der zyklischen bzw. azyklischen Flottenzuweisung bereits für zwei Flotten streng NP-vollständig und nicht approximierbar ist (Tabelle 3.4).

Beim Vergleich der Ergebnisse zwischen zyklischen und azyklischen Problemen fällt auf, dass azyklische Probleme manchmal leichter zu lösen sind. Während zum Beispiel die Klasse FAP$(Opt, Cy, 1, Ar, Cdt, Cdp, Big)$ NP-vollständig, ist lässt die azyklische Variante in Zeit $O(|\mathcal{L}|^3)$ lösen (Tabelle 3.1). Entsprechendes gilt für die Problemklassen FAP$(Feas, Cy, f, Eq)$ und FAP(Inf, Cy, f, Eq) mit mindestens drei Flotten. Allerdings zeigt sich hier, dass sich auch die azyklischen Varianten in ihrer Komplexität unterscheiden können, denn die Klasse FAP$(Feas, Ac2, f, Eq)$ ist im Gegensatz zu den $Ac0$- und $Ac1$-Varianten streng NP-vollständig (Tabelle 3.3).

Die jetzt noch offenen Punkte betreffen fast ausschließlich Klassen mit zwei Flotten, die über dieselben operationellen Eigenschaften verfügen. Bereits kleine Erweiterungen wie verbindungsabhängige Gewinne oder sehr lange Blockzeiten führen dazu, das die Probleme NP-vollständig werden. Aber es ist unklar, ob das auch für die nicht-erweiterten Klassen gilt. Schließlich stellt sich für die Klassen FAP$(Opt, Ac0, f, Eq)$ und FAP$(Opt, Ac1, f, Eq)$ die Frage, ob die Probleme in irgend einer Form in polynomieller Zeit approximierbar sind. Dies sind die einzigen Opt-Klassen, für die das korrespondierende Zulässigkeitsproblem in polynomieller Zeit gelöst werden kann (Tabelle 3.3).

— 4 —

Lösungsverfahren

In diesem Kapitel stellen wir verschiedene neu entwickelte exakte und heuristische Lösungsverfahren für das Flottenzuweisungsproblem vor, die sowohl für die zyklische als auch azyklische Variante geeignet sind. Wir starten mit einer Beschreibung von einfachen Transformationen, mit denen sich azyklische in zyklische Flottenzuweisungsprobleme umwandeln lassen. Als nächstes präsentieren wir das von [Hane et al., 1995] eingeführte Time Space Network Modell, eine IP-Formulierung des Flottenzuweisungsproblems, mit der sich auch große Instanzen lösen lassen. Dieses Modell bildet die Basis für unsere exakten Lösungsverfahren.

In den Abschnitten 4.3 und 4.4 folgt die Beschreibung unserer Lokale Suche Verfahren, die einen Hill Climbing- bzw. Simulated Annealing-Ansatz verfolgen. Das zentrale Thema ist die dabei zum Einsatz kommende problemspezifische Nachbarschaft für das Flottenzuweisungsproblem. Anschließend zeigen wir, wie zwei für die Praxis wichtige Erweiterungen für das Flottenzuweisungsproblem, verbindungsabhängige Mindestbodenzeiten/Gewinne und die so genannte Homogenität einer Zuweisung, berücksichtigt werden können. Wir modifizieren dafür unsere Heuristiken und präsentieren neue IP-Formulierungen.

Es folgt die Beschreibung von Preprocessing-Techniken, mit deren Hilfe sich zum einen vorab die Zulässigkeit einer Instanz abschätzen lässt und die zum anderen die Eingabegröße einer Instanz verkleinern können. Insbesondere die exakten Ansätze profitieren davon. Als letztes evaluieren wir die in diesem Kapitel vorgestellten Lösungsverfahren, Erweiterungen und Preprocessing-Techniken mittels realer Instanzen, die aus den Planungsabteilungen verschiedener Fluggesellschaften stammen.

4.1 Elementare Transformationen

Die in Kapitel 2 definierten zyklischen und azyklischen Flottenzuweisungsprobleme sind sehr allgemein gehalten. Sie unterstützen Spezialitäten wie

- zyklische und verschiedene azyklische Varianten
- eingeschränkte Auswahl an möglichen Flotten für ein Leg
- flottenabhängige Block- und Startzeiten
- verbindungsabhängige Mindestbodenzeiten und Gewinne
- Block- und Mindestbodenzeiten, die länger als die Planungsperiode sind

In diesem Kapitel werden wir noch weitere praxisrelevante Erweiterungen vorstellen.

Ein in der Praxis einsetzbares Lösungsverfahren für die Flottenzuweisung muss nicht notwendigerweise alle diese Spezialitäten unterstützen, da sie, abhängig vom Verfahren, nicht oder nur sehr schwer zu implementieren sind. Das Flottenzuweisungsproblem mit den meisten Einschränkungen, das noch als solches anerkannt wird, ist die Klasse FAP(Opt, Cy, f, Eq) aus Kapitel 3. Keine der oben angegebenen Spezialitäten wird von ihr direkt unterstützt.

Flottenzuweisungsprobleme mit manchen Spezialitäten lassen sich allerdings trotzdem durch ein Verfahren lösen, das diese Spezialitäten eigentlich gar nicht unterstützt. Ein einfaches Beispiel dafür ist die eingeschränkte Auswahl an möglichen Flotten für ein Leg. Unterstützt ein Lösungsverfahren dies nicht, können trotzdem Instanzen mit dieser Spezialität gelöst werden, indem nicht-zulässige Zuweisungen mit hohen Strafkosten belegt werden. Optimal arbeitende Lösungsverfahren werden solche Zuweisungen dann nicht verwenden. Werden eigentlich nicht unterstützte Spezialitäten wie in diesem Beispiel über die Zielfunktion modelliert, kann es allerdings bei Heuristiken dazu führen, dass diese eine eigentlich unzulässige Lösung produzieren. Ferner ist es häufig so, dass einen direkte Unterstützung von Spezialitäten, sofern möglich, zu deutlich effizienteren Algorithmen führt.

Wie der nun folgende Satz zeigt, reicht es aus, ein Lösungsverfahren für das zyklische Flottenzuweisungsproblem zu besitzen. Die azyklischen Varianten $Ac1$ und $Ac2$ lassen sich sehr effizient auf den zyklischen Fall reduzieren. Möglich ist dies auch für die azyklische Variante $Ac0$. Allerdings benötigt man hier eine zusätzliche Flotte.

Satz 4.1. *Es gilt:*

1. FAP$(Opt, Ac2, f, *) =_p$ FAP$(Opt, Ac1, f, *) \leq_p$ FAP$(Opt, Ac0, f, *)$
2. FAP$(Opt, Ac2, f, *) =_p$ FAP$(Opt, Ac1, f, *) \leq_p$ FAP$(Opt, Cy, f, *)$
3. FAP$(Opt, Ac0, f, *) \leq_p$ FAP$(Opt, Cy, f + 1, *)$

„$\leq_p$“ steht dabei für „(einfach) in polynomieller Zeit reduzierbar“.

Beweis. Wir werden in diesem Beweis häufig Legs verwenden, die nur von einer Auswahl der Flotten bedient werden können. Unterstützt die betrachtete Problemklasse dies nicht direkt, lässt sich dies, wie weiter oben beschrieben, über die Zielfunktion simulieren. Für eine azyklische Flottenzuweisungsinstanz sei T die späteste Ankunftszeit aller Legs.

FAP$(Opt, Ac2, f, *) \leq_p$ FAP$(Opt, Ac1, f, *)$:

Wir führen einen zusätzlichen Flughafen XYZ ein. Für jeden Flughafen s und jede Flotte f starten von Flughafen s zum Zeitpunkt T insgesamt $N^e_{s,f}$ Legs zum Flughafen XYZ, die nur von Flotte f bedient werden dürfen. Die Blockzeit aller dieser Legs sei 1. Wir können nun die Flugzeugverteilung zum Periodenende weglassen, da nach Konstruktion sichergestellt ist, dass sich nun am Periodenende alle Flugzeuge auf Flughafen XYZ befinden müssen und vorher in der passenden Anzahl $N^e_{s,f}$ von den jeweiligen Flughäfen gekommen sind.

FAP$(Opt, Ac1, f, *) \leq_p$ FAP$(Opt, Ac2, f, *)$:

Wir beschreiben hier nur den Fall, dass für eine Instanz aus FAP$(Opt, Ac1, f, *)$ die Verteilung der Flugzeuge zu Periodenbeginn vorgegeben ist. Der Fall, bei dem die Verteilung am Periodenende vorgegeben ist, verläuft ähnlich.

Mit der Verteilung der Flugzeuge zu Periodenbeginn ist für eine Instanz auch bekannt, wie viele Flugzeuge sich am Ende der Planungsperiode auf jedem Flughafen befinden.[1] Auf einem Flughafen s mit D_s startenden und A_s ankommenden Legs müssen sich bei einer zulässigen Flottenzuweisung am Periodenende

$$E_s = \sum_{f \in \mathcal{F}} N^b_{s,f} + A_s - D_s$$

Flugzeuge auf Flughafen s aufhalten.

Wir führen nun einen neuen Flughafen XYZ ein. Von jedem Flughafen s starten zum Zeitpunkt T insgesamt E_s neue Legs zum Flughafen XYZ mit Blockzeit 1. Damit wird offensichtlich sichergestellt, dass eine zulässige Zuweisung der transformierten Instanz die Flugzeuge aller Flotten zum Periodenende auf Flughafen XYZ versammelt, und wir können die Verteilung der Flugzeuge zum Periodenende angeben:

$$N^e_{XYZ,f} = \sum_{s \in \mathcal{S}} N^b_{s,f}$$

FAP$(Opt, Ac1, f, *) \leq_p$ FAP$(Opt, Ac0, f, *)$:

Wir verfahren hier analog zu dem Fall FAP$(Opt, Ac2, f, *) \leq_p$ FAP$(Opt, Ac1, f, *)$, um die Verteilung der Flugzeuge am Periodenende zu eliminieren. Die Gesamtflugzeuganzahl der einzelnen Flotten berechnet sich wie folgt:

$$N_f = \sum_{s \in \mathcal{S}} N^e_{s,f}$$

FAP$(Opt, Ac2, f, *) \leq_p$ FAP$(Opt, Cy, f, *)$:

Wir eliminieren die Verteilung der Flugzeuge zu Periodenbeginn und zum Periodenende wie im Fall FAP$(Opt, Ac2, f, *) \leq_p$ FAP$(Opt, Ac1, f, *)$ und benutzen in beiden Fällen den selben zusätzlichen Flughafen XYZ. Die Gesamtflugzeuganzahl der einzelnen Flotten berechnet

[1] Nur die Gesamtanzahl über alle Flotte ist bekannt!

sich durch $N_f = \sum_{s \in \mathcal{S}} N^e_{s,f}$. Jeder Flughafen ist nun balanciert und auf Flughafen XYZ müssen sich sowohl zum Periodenbeginn als auch zum Periodenende alle Flugzeuge aufhalten. Damit ist jede zulässige Lösung auch automatisch zyklisch und wir können die Periodenlänge $\mathcal{T}$ der zyklischen Instanz auf $T + 2$ festsetzen.

FAP$(Opt, Ac0, f, *) \leq_p$ FAP$(Opt, Cy, f + 1, *)$:

Das Problem hier ist, dass wir für die einzelnen Flughäfen nicht wissen, wie viele Flugzeuge dort insgesamt zum Periodenanfang bzw. -ende warten. Die zyklische Instanz muss in der Lage sein, genügend Flugzeuge auf allen Flughäfen zur Verfügung zu stellen, ohne genau zu wissen wie viele das sind.

Dazu führen wir wieder einen zusätzlichen Flughafen XYZ ein, von dem für jeden Flughafen s zu Periodenbeginn D_s Legs nach s aufbrechen. Zum Periodenende kommen von jedem Flughafen s A_s Legs nach XYZ zurück. D_s und A_s bezeichnen dabei wieder die Anzahl an Starts bzw. Landungen, die auf Flughafen s stattfinden. Im worst-case benötigt jedes startenden Leg ein eigenes Flugzeug, so dass die so zur Verfügung gestellten neuen Legs ausreichen, um genügend Flugzeuge auf jedem Flughafen zur Verfügung zu stellen. Damit starten und landen auf XYZ jeweils $|\mathcal{L}|$ Legs und alle Flughäfen sind balanciert.

Nicht alle neuen Legs können aber von den verfügbaren Flugzeugen gleichzeitig bedient werden. Daher führen wir eine zusätzliche Flotte mit $|\mathcal{L}|$ vielen Flugzeugen ein, die ausschließlich die neu hinzugefügten Legs bedienen kann. Eigentlich nicht benötigte neue Legs werden von der neuen Flotte bedient, die ansonsten den restlichen Flugplan nicht weiter beeinflussen kann.

Es ist nun nicht schwer zu zeigen, dass die transformierte zyklische Instanz genau dann eine Lösung mit Gewinn G besitzt, wenn dies auch für die azyklische Ursprungsinstanz zutrifft. Dabei sind die Flottenzuweisungen der beiden Lösungen bezüglich des ursprünglichen Flugplans identisch. □

Bemerkung 4.2. *Nach den Ergebnissen aus Kapitel 3 ist klar, dass FAP$(Opt, t, f, *) =_p$ FAP$(Opt, t', f', *')$ für $t, t' \in \{Cy, Ac, Ac0, Ac1, Ac2\}$ und $f, f' \geq 3$ gilt, da all diese Klassen NP-vollständig sind. Allerdings sind die dafür notwendigen Reduktionen sehr wahrscheinlich nicht so einfach wie die aus Satz 4.1.*

4.2 Time Space Network

Liegt für eine Problemklasse P eine Formulierung als (gemischt-ganzzahliges) lineares Programm vor, liefern die etablierten Lösungsverfahren der linearen bzw. ganzzahligen linearen Programmierung direkt ein exaktes Lösungsverfahren für P. In Kapitel 2 haben wir das Flottenzuweisungsproblem mittels ganzzahliger linearer Programme (Modelle 2.4 und 2.7) definiert, so dass wir damit prinzipiell bereits ein Lösungsverfahren für das Flottenzuweisungsproblem angegeben haben. Allerdings haben wir bereits in Kapitel 2 angemerkt, dass sich diese Modelle wegen ihrer Größe in der Praxis nicht bewährt haben.

Für eine etwas eingeschränktere Klasse von Flottenzuweisungsproblemen definieren wir in diesem Abschnitt nun ein weiteres gemischt-ganzzahliges lineares Modell. Dabei darf das Flottenzuweisungsproblem

- keine verbindungsabhängigen Mindestbodenzeiten und
- keine verbindungsabhängigen Gewinne

beinhalten. Daraus folgt, dass die Ankunftszeit eines Legs nur von der Flotte abhängig ist, die dieses Leg bedient (und nicht auch noch vom Folgeleg). In Abschnitt 4.5 stellen wir ein von uns neu entwickeltes Modell vor, das auch verbindungsabhängige Mindestbodenzeiten und verbindungsabhängige Gewinne unterstützt.

Das im Folgenden beschriebene Modell, das so genannte *Time Space Network Modell*, ist in [Hane et al., 1995] für zyklische Flottenzuweisungsprobleme eingeführt worden und kann als das in der Praxis am häufigsten eingesetzte Verfahren zur Flottenzuweisung im Flugverkehr angesehen werden. Wie bei den Connection Network Modellen aus Kapitel 2 handelt es sich bei dem Time Space Network Modell um die IP-Formulierung eines erweiterten ganzzahligen (Mehrgüter-)Flussproblems.

4.2.1 Zyklisches Modell

Das dem Modell zugrunde liegende gerichtete Flussnetzwerk $G = (V, E)$, *Time Space Network* genannt, besteht aus folgenden Knoten:

$$V = \left\{(s_l^d, f, t_{l,f}^d) \in \mathcal{S} \times \mathcal{F} \times [\mathcal{T}] \mid l \in \mathcal{L} \wedge f \in \mathcal{F}_l\right\} \cup \left\{(s_l^a, f, t_{l,f}^a) \in \mathcal{S} \times \mathcal{F} \times [\mathcal{T}] \mid l \in \mathcal{L} \wedge f \in \mathcal{F}_l\right\}$$

Bei V handelt es sich um die Menge aller *möglichen* Flugereignisse, die in einer Flottenzuweisung auftreten können. Ein Flugereignis $(s, f, t) \in \mathcal{S} \times \mathcal{F} \times [\mathcal{T}]$ beschreibt dabei den Zeitpunkt t eines möglichen Starts oder einer möglichen Ankunft auf Flughafen s von einem Flugzeug der Flotte f.

Zu einem Ereignis $v \in V$ definieren wir wie folgt die zu diesem Ereignis gehörenden ankommenden ($\mathcal{L}_v^a$) und abfliegenden ($\mathcal{L}_v^d$) Leg-Flotten-Kombinationen:

$$\mathcal{L}_v^a = \left\{(l, f) \in \mathcal{L} \times \mathcal{F} \mid v = (s_l^a, f, t_{l,f}^a)\right\}$$
$$\mathcal{L}_v^d = \left\{(l, f) \in \mathcal{L} \times \mathcal{F} \mid v = (s_l^d, f, t_{l,f}^d)\right\}$$

Wir partitionieren die Knotenmenge in Teilmengen von Ereignissen, die auf dem selben Flughafen $s \in \mathcal{S}$ für die selbe Flotte $f \in \mathcal{F}$ stattfinden:

$$V_{s,f} = V \cap \{s\} \times \{f\} \times [\mathcal{T}]$$

Seien $\{v_0, \ldots, v_{n-1}\} = V_{s,f}$ die Knoten von $V_{s,f}$ aufsteigend sortiert nach den Ereigniszeitpunkten. Wir definieren zu jedem Knoten $v = v_i \in V_{s,f}$ seinen Vorgänger $v^- = v_{(i-1) \bmod n} \in V_{s,f}$ und seinen Nachfolger $v^+ = v_{(i+1) \bmod n} \in V_{s,f}$. Ferner bezeichnen wir mit $v_{s,f}^{\min} = v_0$ das frühste Ereignis in $V_{s,f}$ und mit $v_{s,f}^{\max} = v_{n-1}$ das späteste Ereignis.

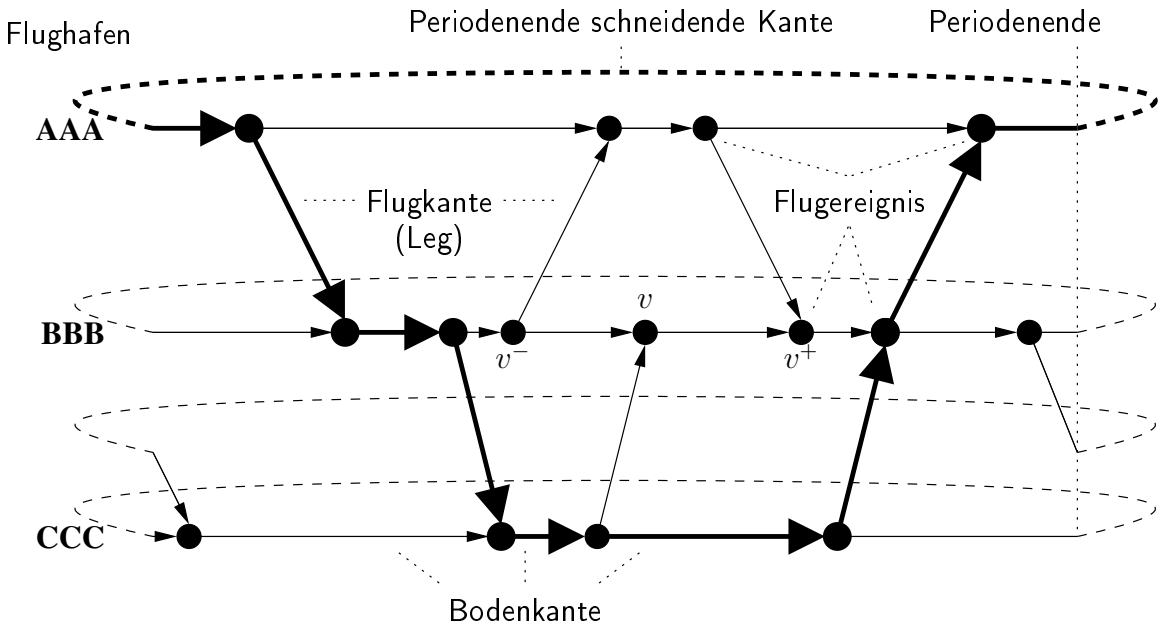


Abbildung 4.1: Ausschnitt aus einem Time Space Network (für eine Flotte f). Die fetten Kanten markieren einen möglichen Flugzeugumlauf.

Die Kantenmenge E des Time Space Networks

$$E = \left\{ \left((s_l^d, f, t_{l,f}^d), (s_l^a, f, t_{l,f}^a) \right) \mid l \in \mathcal{L} \wedge f \in \mathcal{F}_l \right\} \cup \left\{ (v, v^+) \mid v \in V \right\}$$

besteht aus zwei Arten von Kanten. Die erste Art wird *Flugkanten* genannt, und verbindet das Startereignis eines Legs l, wenn es von Flotte f bedient wird, mit seinem Ankunftsereignis. Die zweite Art von Kanten sind die *Bodenkanten*, die die Ereignisse jeder $V_{s,f}$-Teilmenge zyklisch miteinander verbinden.

Abbildung 4.1 zeigt den Ausschnitt eines Time Space Networks für eine Flotte f. Für jede weitere Flotte kommt ein ähnliches Netzwerk zum Gesamtnetzwerk hinzu. Wegen unterschiedlicher Start- und Blockzeiten und etwaigen Flotteneinschränkungen an Legs können sich die Teilnetzwerke der einzelnen Flotten topologisch durchaus voneinander unterscheiden.

Die Güter, die im Time Space Network verschickt werden können, repräsentieren die Flugzeuge der einzelnen Flotten. Ein Fluss von 1 auf einer Flugkante symbolisiert, dass das zugehörige Leg von der zugehörigen Flotte bedient wird. Bodenkanten überführen auf einem Flughafen ankommende Flugzeuge zu ihrem nächsten Abflugereignis. Da sich auf einem Flughafen zu einem Zeitpunkt mehrere Flugzeuge aufhalten dürfen, ist die Kapazität der Bodenkanten unbeschränkt.

Eine Konsequenz daraus ist, dass ein Time Space Network Modell *keine* Verknüpfung zwischen ankommenden und abfliegenden Legs berechnet. Wenn über eine Bodenkante ein Fluss von zwei fließt und am Endknoten ein Flugzeug den Flughafen verlässt, legt das Time Space

Network Modell nicht fest, welches der beiden Flugzeuge starten soll. Bodenkanten anonymisieren daher auf einem Flughafen ankommende Flugzeuge.

Die Anzahl der durch einen Fluss im Time Space Network verbrauchten Flugzeuge bestimmen wir wie im Fall des Connection Networks, indem wir den Gesamtfluss je Flotte über einen Schnitt zum Periodenbeginn messen. Dazu definieren wir für jede Flotte $f \in \mathcal{F}$

$$V_f^\Delta = \bigcup_{s \in \mathcal{S}} \{v_{s,f}^{\min}\}$$

als die Menge der frühsten Ereignisse einer Flotte. Bodenkanten zu Vorgängern dieser Ereignisse führen über den Schnitt am Periodenanfang. Ferner verbraucht ein Leg $l \in \mathcal{L}$, wenn es von Flotte $f \in \mathcal{F}_l$ bedient wird, ähnlich wie im Connection Network

$$\Delta_{l,f} = \left\lfloor \frac{b_{l,f} + g_{l,f}}{\mathcal{T}} \right\rfloor + \rho(t_{l,f}^d, t_{l,f}^a)$$

viele Flugzeuge.

Das Modell verwendet zwei Klassen von Variablen, die den Fluss auf den Kanten des Time Space Networks repräsentieren:

$y_{l,f}$	Boolsche Variable, die anzeigt, ob das Leg l von einem Flugzeug der Flotte $f \in \mathcal{F}$ bedient werden soll ($y_{l,f} = 1$) oder nicht ($y_{l,f} = 0$)
z_{v,v^+}	Anzahl zwischen den Ereignissen $v = (s, f, t)$ und v^+ auf dem Flughafen s wartenden Flugzeuge von Flotte f

Das zyklische Flottenzuweisungsproblem lässt sich damit wie folgt formulieren:

Modell 4.3 (Zyklisches Time Space Network).

$$\textit{Maximiere} \sum_{l \in \mathcal{L}} \sum_{f \in \mathcal{F}_l} p_{l,f} y_{l,f} \tag{4.1}$$

unter den Nebenbedingungen

$$\sum_{f \in \mathcal{F}_l} y_{l,f} = 1 \qquad \forall l \in \mathcal{L} \tag{4.2}$$

$$\sum_{(l,f) \in \mathcal{L}_v^a} y_{l,f} - \sum_{(l,f) \in \mathcal{L}_v^d} y_{l,f} + z_{v^-,v} - z_{v,v^+} = 0 \qquad \forall v \in V \tag{4.3}$$

$$\sum_{l \in \mathcal{L}: f \in \mathcal{F}_l} \Delta_{l,f} y_{l,f} + \sum_{v \in V_f^\Delta} z_{v^-,v} \leq N_f \qquad \forall f \in \mathcal{F} \tag{4.4}$$

$$y_{l,f} \in \{0,1\} \qquad \forall l \in \mathcal{L}, f \in \mathcal{F}_l \tag{4.5}$$

$$z_{v,v^+} \in \mathbb{N}_0 \qquad \forall v \in V \tag{4.6}$$

Die Zielfunktion (4.1) maximiert den Gesamtgewinn der Flottenzuweisung. Die Gleichungen (4.2) garantieren, dass jedes Leg von genau einer Flotte bedient wird. Hierüber werden die aus Graphensicht unabhängigen Teilnetzwerke je Flotte miteinander gekoppelt. Die Gleichungen (4.3) sind die normalen Flusserhaltungsgleichungen für jedes Ereignis.

Die Ungleichungen (4.4) beschränken für jede Flotte die Anzahl verwendeter Flugzeuge. Dazu wird für jede Flotte der Gesamtfluss „über einen Schnitt zum Zeitpunkt Null“ bestimmt. Dieser setzt sich aus den sich zu diesem Zeitpunkt in der Luft befindlichen Flugzeugen (erste Summe) und den auf den einzelnen Flughäfen wartenden Flugzeugen (zweite Summe) zusammen.

Die Bedingungen (4.5) definieren die verwendeten Flugkanten-Variablen als Boolsch und die Bedingungen (4.6) definieren die Bodenkanten-Variablen als nicht-negativ. Alternativ zu den Bedingungen (4.6) reicht es dabei, die Bodenkanten-Variablen als

$$z_{v,v^+} \geq 0 \qquad \forall v \in V \tag{4.7}$$

zu definieren, da etwaige fraktionale Anteile nur Teil eines zyklischen fraktionalen Flusses auf den Bodenkanten eines Flughafens sein können. Ein solcher Teilfluss verbraucht aber nur entsprechend viele Flugzeuge und stört die eigentliche Flottenzuweisung an Legs nicht weiter.

Die Menge der Flugereignisse V hat eine Größe kleiner gleich $2|\mathcal{L}| \cdot |\mathcal{F}|$, da jedes Leg-Flotten-Paar zu zwei Ereignissen gehört. Entsprechend viele Bodenkanten gibt es. In Abschnitt 4.7.2 werden wir zeigen, dass sich diese Anzahl durch ein einfaches Preprocessing immer mindestens zu $|\mathcal{L}| \cdot |\mathcal{F}|$ machen lässt. Daraus folgt:

Beobachtung 4.4 (Größe des Time Space Network Modells). *Eine Flottenzuweisungsinstanz mit Legmenge $\mathcal{L}$ und Flottenmenge $\mathcal{F}$ erzeugt ein Time Space Network Modell mit $O(|\mathcal{F}| \cdot |\mathcal{L}|)$ vielen Variablen und $O(|\mathcal{F}| \cdot |\mathcal{L}|)$ Nebenbedingungen. Offensichtlich lässt sich ein entsprechendes Modell in polynomieller Zeit erzeugen.*

Der große Vorteil des Time Space Network Modells im Vergleich zum Connection Network Modell ist, dass seine Variablenanzahl nur linear mit der Leg- und Flottenanzahl wächst. Dies macht das Modell selbst für große Instanzen mit tausenden von Legs lösbar.

4.2.2 Azyklisches Modell

Das Time Space Network Modell von [Hane et al., 1995] lässt sich einfach an das azyklische Flottenzuweisungsproblem anpassen:

- Die Knotenmenge im zugrunde liegenden Flussnetzwerk bleibt unverändert.
- Aus der Kantenmenge werden all die Bodenkanten entfernt, die den Periodenbeginn enthalten, das heißt, alle Kanten $(v_{s,f}^{\max}, v_{s,f}^{\min})$ $(s \in \mathcal{S},\ f \in \mathcal{F})$.
- Der Vorgänger des frühsten Ereignisses eines jeden Flughafens wird auf das Hilfsereignis $*$ gesetzt: $(v_{s,f}^{\min})^- = *$
- Der Nachfolger des spätesten Ereignisses eines jeden Flughafens wird ebenfalls auf das Hilfsereignis $*$ gesetzt: $(v_{s,f}^{\min})^+ = *$

Das azyklische Time Space Network Modell lässt sich damit wie folgt formulieren:

Modell 4.5 (Azyklisches Time Space Network).

$$\textit{Maximiere} \sum_{l \in \mathcal{L}} \sum_{f \in \mathcal{F}_l} p_{l,f} y_{l,f} \tag{4.8}$$

unter den Nebenbedingungen

$$\sum_{f \in \mathcal{F}_l} y_{l,f} = 1 \qquad \forall l \in \mathcal{L} \tag{4.9}$$

$$\sum_{(l,f) \in \mathcal{L}_v^a} y_{l,f} - \sum_{(l,f) \in \mathcal{L}_v^d} y_{l,f} + z_{v^-,v} - z_{v,v^+} = 0 \qquad \forall v \in V \tag{4.10}$$

$$\sum_{v \in V_f^\Delta} z_{*,v} \leq N_f \qquad \forall f \in \mathcal{F} \tag{4.11}$$

$$y_{l,f} \in \{0,1\} \qquad \forall l \in \mathcal{L}, f \in \mathcal{F}_l \tag{4.12}$$

$$z_{v,v^+} \in \mathbb{N}_0 \qquad \forall v \in V \tag{4.13}$$

$$z_{*,v} \in \mathbb{N}_0 \qquad \forall v \in \bigcup_{f \in \mathcal{F}} V_f^\Delta \tag{4.14}$$

Das zyklische und das azyklische Time Space Network Modell unterscheiden sich praktisch nur in den Ungleichungen (4.11). Im azyklischen Modell wird hier einfach nur der Fluss in jeden Flughafen aufsummiert, da sich im azyklischen Fall zum Periodenbeginn keine Flugzeuge in der Luft befinden können.

Durch eine passende Fixierung der $z_{*,v}$- bzw. $z_{v,*}$-Variablen lassen sich alternativ zu den Ungleichungen (4.11) die Flugzeugverteilungen der Flotten auf den einzelnen Flughäfen zu Periodenbeginn bzw. zum Periodenende vorgeben.

4.3 Lokale Suche Heuristiken

In diesem Abschnitt präsentieren wir von uns entwickelte Lokale Suche Heuristiken für das Flottenzuweisungsproblem: ein Hill Climbing Verfahren und ein Simulated Annealing Verfahren. Beide Algorithmen verwenden dabei dieselbe Nachbarschaft.

Die Verfahren werden seit einigen Jahren von mehr als zehn internationalen Fluggesellschaften für ihre Flotteneinsatzplanung verwendet. Dabei haben sie sich auch in der Praxis bewährt, indem sie zuverlässig profitable Flottenzuweisungen in kurzer Zeit berechnen konnten.

Die in Abschnitt 4.3.1 beschriebene Nachbarschaft spielt dabei die zentrale Rolle für die Heuristiken. Im Vergleich zu vielen anderen Optimierungsproblemen ist es beim Flottenzuweisungsproblem bereits schwierig,[2] überhaupt eine zulässige Lösung zu finden. Auch das für eine Lokale Suche Heuristik essentielle Ändern einer existierenden Lösung führt beim Flottenzuweisungsproblem leicht dazu, dass die neue Lösung unzulässig wird. Daher ist die hier zum Einsatz kommenden Nachbarschaft recht komplex, vor allem im Vergleich zu anderen

[2] genauer gesagt NP-vollständig, siehe Kapitel 3

Optimierungsproblemen wie dem Traveling Salesman Problem, aber dennoch effizient. Unsere Nachbarschaft garantiert, dass eine zulässige Lösung beim Nachbarschaftsübergang nicht unzulässig wird, und ist auf der anderen Seite ausdrucksvoll genug, um den Lösungsraum effektiv durchsuchen zu können.

Wie auch schon das in Abschnitt 4.2 vorgestellte Time Space Network Modell unterstützen unsere Heuristiken nicht alle Besonderheiten des in Kapitel 2 definierten Flottenzuweisungsproblems. Es darf

- keine verbindungsabhängigen Mindestbodenzeiten,
- keine verbindungsabhängigen Gewinne,
- keine flottenabhängigen Startzeiten und
- keine Block- und Mindestbodenzeiten, die länger als die Planungsperiode sind,

beinhalten. Flottenabhängige Startzeiten und lange Block- und Mindestbodenzeiten treten bei Flottenzuweisungsproblemen in der Praxis sehr selten auf. Damit können die hier beschriebenen Heuristiken aus Praxissicht dieselben Probleminstanzen optimieren, wie das Time Space Network Modell aus Abschnitt 4.2. Wie man die Heuristiken erweitert, um verbindungsabhängige Mindestbodenzeiten zu unterstützen, beschreiben wir in Abschnitt 4.5.

Eine wichtige Einschränkung unserer Heuristiken ist, dass

- eine zulässige Flottenzuweisung für die Eingabeinstanz

bekannt sein muss.[3] Je nach Anwendungsfall ist diese Einschränkung unproblematisch bis nicht-akzeptabel. In der strategischen Planung wird halbjährlich die grobe Flotteneinsatzplanung für die nächste Saison durchgeführt. Dabei ändern sich die Flugpläne für die betrachteten Standardplanungsperioden typischerweise von Mal zu Mal kaum. In diesem Fall kann man also auf die Lösung der vorherigen Planungsperiode zurückgreifen. Auf der anderen Seite muss im Störungsmanagement auf unvorhergesehene Situationen reagiert werden, die die aktuelle Planung über den Haufen werfen. Hier ist häufig keine zulässige Lösung bekannt, da die Hauptaufgabe des Störungsmanagements gerade das Wiederherstellen eines zulässigen Plans ist.

Ein großer Vorteil von Heuristiken im Allgemeinen ist, dass sie meistens nicht darauf angewiesen sind, dass sich die zu lösende Problemstellung gut linearisieren lässt, also in Form von linearen Ungleichungen und einer linearen Zielfunktion ausdrücken lässt. In Abschnitt 4.6 und Kapitel 6 sieht man, dass hier unsere Heuristiken linearen Programmen überlegen sind.

Wir beschreiben im Folgenden nur die Vorgehensweise für zyklische Flottenzuweisungsprobleme. Die Verfahren sind aber auch für die azyklischen Varianten angepasst und implementiert worden. Die Details sind eher technischer Natur und werden in dieser Arbeit daher nicht beschrieben.

[3] Die Flugzeuganzahlen dürfen dabei durchaus überschritten werden. Allerdings garantieren die Verfahren dann nicht, dass sie überhaupt eine zulässige Lösung zurückliefern.

4.3.1 Swap-Change-Nachbarschaft

Das Verändern einer Lösung eines Flottenzuweisungsproblems geschieht dadurch, dass die Flottenzuweisung von einigen Legs verändert wird. Beim Verändern einer gegebenen zulässigen Lösung für das zyklische Flottenzuweisungsproblem müssen dabei zwei Punkte besonders beachtet werden:

- Die neue Lösung muss weiterhin bezüglich der Starts und Landungen der einzelnen Flotten auf den Flughäfen balanciert sein.
- Die vorgegebenen Flugzeuganzahlen je Flotte dürfen nicht überschritten werden.

Dies sind die beiden zentralen Eigenschaften, die die Komplexität des Flottenzuweisungsproblems ausmachen.

4.3.1.1 Balanciertheit

Das Ändern der Flottenzuweisung nur eines Legs führt automatisch dazu, dass die neue Zuweisung unbalanciert ist. Also muss bei einem Nachbarschaftübergang die Flottenzuweisung für eine Menge von Legs geändert werden, um die Zulässigkeit zu erhalten. Wir definieren dazu den Begriff der Legfolge.

Definition 4.6 (Legfolge). *Gegeben sei eine zulässige Zuweisung $z : \mathcal{L} \to \mathcal{F}$, auch* aktuelle Lösung *genannt, einer Flottenzuweisungsinstanz.*

Eine nicht-leere Folge von Legs $F = (l_1, \ldots, l_n)$, $l_i \in \mathcal{L}$, heißt Legfolge (mit n Legs), *wenn gilt:*

$$\begin{aligned} l_i &\neq l_j & \forall i \neq j \\ z(l_i) &= z(l_j) & \forall i, j \in \{1, \ldots, n\} \\ s^a_{l_i} &= s^d_{l_{i+1}} & \forall i \in \{1, \ldots, n-1\} \end{aligned}$$

Wir definieren $|F| = n$ als die Länge der Legfolge, $s^d_F = s^d_{l_1}$ als den Startflughafen der Legfolge und $s^a_F = s^a_{l_n}$ als deren Zielflughafen. $F_i = l_i$ bezeichnet das i-te Leg der Legfolge und $z(F) = z(l_1)$ die der Legfolge zugewiesene Flotte.

Die Legfolge heißt balanciert, *wenn $s^d_F = s^a_F$ gilt.*

Mit dem Zuweisen einer Flotte f an eine Legfolge F *bezeichnen wir Vorgang, die Flottenzuweisung aller Legs der Legfolge auf f zu ändern.*

Beobachtung 4.7. *Durch das Zuweisen einer neuen Flotte f an eine Legfolge F ist garantiert, dass auf allen Flughäfen bis auf den Start- und Zielflughafen der Legfolge die flottenweise Balancierung der Zuweisung erhalten bleibt.*

Die Nachbarschaft unserer Heuristiken ist durch zwei Arten von Nachbarschaftübergängen definiert. Der eine Übergang weist einer balancierten Legfolge eine neue Flotte zu und wird *Change-Übergang* genannt. Der zweite Übergang tauscht die Flottenzuweisung zweier Legfolgen gegeneinander aus und wird *Swap-Übergang* genannt.

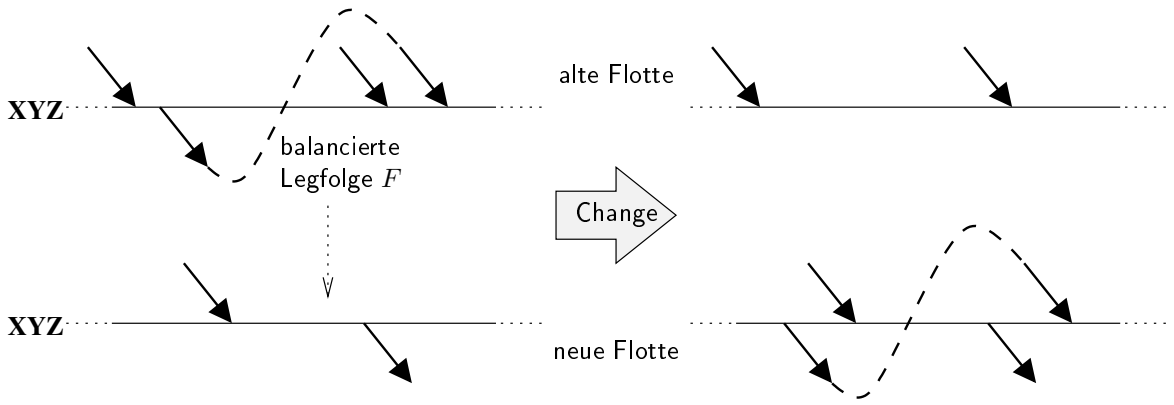


Abbildung 4.2: Change-Übergang

Change-Übergang Sind bei einer Legfolge F der Start- und Zielflughafen identisch ($s_F^d = s_F^a$), so heißt die Legfolge *balanciert*. Dieser Flughafen wird auch kurz der *Flughafen der (balancierten) Legfolge* F genannt.

Weist man nun einer balancierten Legfolge eine neue Flotte zu, so bleibt die Balanciertheit des Flugplans erhalten; auf dem Startflughafen der Legfolge, der gleichzeitig auch der Zielflughafen ist, wird bei der alten Flotte ein Start und eine Landung gelöscht, die bei der neuen Flotte hinzukommen.

Die Zuweisung einer neuen Flotte an eine balancierte Legfolge wird als *Change* bezeichnet (Abbildung 4.2). Ein Change ist der eine Typ von Nachbarschaftsoperation, der in den hier beschriebenen Lokale Suche Algorithmen verwendet wird.

Swap-Übergang Der andere Typ von verwendeter Nachbarschaftsoperation benutzt zwei Legfolgen F und G, deren Flotten ausgetauscht werden, d.h. der Legfolge F wird die Flotte $z(G)$ von Legfolge G und der Legfolge G wird Flotte $z(F)$ zugewiesen. Damit dieser Austausch etwas bewirkt, sollte selbstverständlich $z(F) \neq z(G)$ gelten.

Um bei diesem Austausch die Balanciertheit des Flugplans zu erhalten, müssen die Legfolgen F und G ferner folgendes erfüllen:

$$s_F^d = s_G^d \qquad \text{und} \qquad s_F^a = s_G^a$$

Die Startflughäfen der Legfolgen müssen, ebenso wie die Zielflughäfen, gleich sein. F und G bilden dann ein *balanciertes Legfolgenpaar*.

Durch den Austausch wird auf dem Startflughafen für Flotte $z(F)$ der Start von Legfolge F gelöscht, dafür kommt der Start von Legfolge G hinzu; der Flughafen bleibt balanciert. Der Zielflughafen verliert für Flotte $z(F)$ die Landung von Legfolge F und bekommt dafür die Landung von Folge G, bleibt also ebenfalls balanciert. Analog gilt dieses für den Start- und Zielflughafen für Flotte $z(G)$.

Diese Nachbarschaftsoperation, der Austausch der Flotten eines balancierten Legfolgenpaars, wird von nun an als *Swap* bezeichnet. Abbildung 4.3 stellt den Verlauf eines Swaps nochmals schematisch dar.

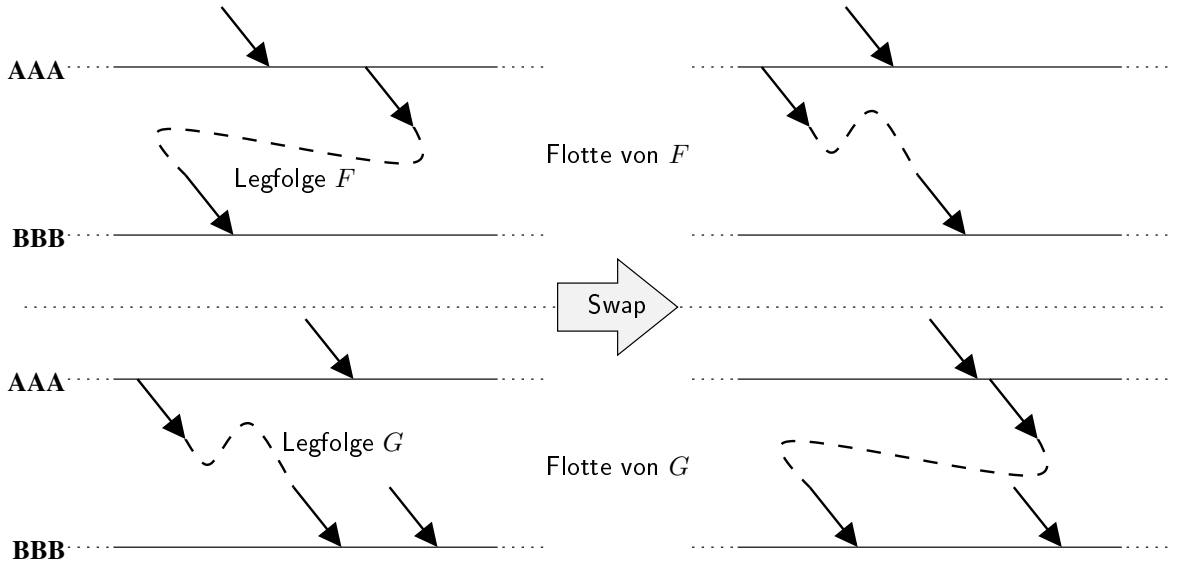


Abbildung 4.3: Swap-Übergang

4.3.1.2 Flugzeuganzahl

Die vorgestellten Nachbarschaftsübergänge Swap und Change stellen bisher nur sicher, dass durch ihre Anwendung die Balanciertheit einer zulässigen Lösung nicht verletzt wird. Dabei sicherzustellen, dass auch die vorgegebenen Flugzeuganzahlen eingehalten werden, ist aufwendiger. Der Trick dabei ist, dass man Legfolgen effizient derart konstruieren kann, dass durch das Zuweisen einer neuen Flotte an eine Legfolge auf jedem von der Legfolge besuchten Flughafen keine *zusätzlichen* Flugzeuge der beiden betroffenden Flotten warten müssen. Damit kann sich die Gesamtflugzeuganzahl je Flotte nicht erhöhen, und eine zulässige Zuweisung bleibt durch einen Change- bzw. Swap-Übergang zulässig.

Wir beschreiben hier zuerst nur, wie die benötigte Flugzeuganzahl einer Zuweisung von den Heuristiken bestimmt wird. Die Details, wie diese Informationen auch dazu genutzt werden können, um bei der Generierung von Legfolgen sicher zu stellen, dass diese keine zusätzlichen Flugzeuge im Fall einer Neuzuweisung erfordern, ist im Detail in Abschnitt 4.4 beschrieben.

Wartefunktion, 0-Zonen und Inseln Die von einer aktuellen Lösung benötigten Flugzeuge der einzelnen Flotten werden wie in Abschnitt 3.3.1 beschrieben mittels der Wartefunktion bestimmt. Darüber hinaus fasst die Wartefunktion die Legs eines Flughafen/Flotten-Paars zu Gruppen, den so genannten *Inseln*, zusammen; alle Legs einer Insel müssen unbedingt untereinander rotationell verknüpft werden, um mit der minimalen Flugzeuganzahl auszukommen ([Gertsbach and Gurevich, 1977]). Zuerst definieren wir dafür:

Definition 4.8 (0-Zone). *Sei W_s^f eine Wartefunktion. Die größtmöglichen Intervalle $[w_S^k, w_E^k)$, in denen die Funktion $W_{sf}(t) = 0, t \in [w_S^k, w_E^k)$ ist und für die während der Zeit (w_S^k, w_E^k) keine Starts und Landungen erfolgen, heißen* 0-Zonen. *$|W_s^f|$ stehe für die Anzahl dieser Intervalle und sie seien mit $k \in \{1, \ldots, |W_s^f|\}$ derart nummeriert, dass $w_S^k < w_S^l$ für $k < l$ gilt.*

Während dieser 0-Zonen stehen auf dem Flughafen also keine Flugzeuge zur Verfügung, und es finden keine Flugereignisse statt. Da jede Wartefunktion mindestens einmal den Wert Null annimmt, existiert für jede Wartefunktion mindestens eine 0-Zone.

Definition 4.9 (Insel). *Sei W_s^f eine Wartefunktion mit $|W_s^f|$ 0-Zonen $[w_S^k, w_E^k)$. Die $|W_s^f|$ Intervalle $[w_E^1, w_S^2]$, $[w_E^2, w_S^3]$, ..., $[w_E^{|W_s^f|}, w_S^1]$ zwischen den 0-Zonen heißen* Inseln.

Im folgenden werden ein paar offensichtliche Eigenschaften von Inseln aufgeführt. Sei $[i_S, i_E]$ eine Insel der Wartefunktion W_s^f:

- Jedes Flugereignis eines balancierten Flugplans kann genau einer Insel zugeordnet werden, zu der es gehört.
- Es gilt $W_s^f(t) > 0$ für $t \in [i_S, i_E)$, das heißt, während einer Insel steht mindestens ein Flugzeug von Flotte f auf Flughafen s zur Verfügung.
- Zum Beginn der Insel (Zeitpunkt i_S) landet (mindestens) ein Flugzeug von Flotte f auf Flughafen s.
- Zum Zeitpunkt i_E startet (mindestens) ein Flugzeug.
- Für zyklische Flottenzuweisungsprobleme gehören zu einer Insel genausoviele Start- wie Landeereignisse, die Insel ist also *balanciert*.

  Im azyklischen Fall gilt dies für alle Inseln bis auf die zeitlich erste und letzte Insel. Wenn zu Periodenbeginn bzw. zum Periodenende Flugzeuge auf dem Flughafen warten, sind die zugehörigen Inseln an der Periodengrenzen nicht balanciert.

Es kann auf einem Flughafen zu einer so genannten *entarteten Insel* kommen. Ist der Landezeitpunkt t_0 eines Legs identisch mit dem Startzeitpunkt eines anderen und fallen beide Ereignisse in eine Phase, zu der kein Flugzeug auf dem Flughafen verfügbar ist, bleibt die Wartefunktion zum Zeitpunkt t_0 gleich Null. Trotzdem entsteht eine Insel $[t_0, t_0]$, eine entartete Insel mit Dauer Null.

Abbildung 4.4 zeigt die Starts und Landungen einer Flotte f auf Flughafen XYZ und die zugehörige Wartefunktion. Desweiteren sind die resultierenden 0-Zonen und Inseln dargestellt.

4.3.1.3 Generieren eines Nachbarschaftsübergangs

Sowohl Swaps als auch Changes bedienen sich Legfolgen, deren Flottenzuweisung geändert wird. Diese Legfolgen werden mittels einer eingeschränkten Tiefensuche generiert, Dies geschieht dynamisch während der Suche nach einem Nachbarn im Lokale Suche Optimierer. Ausgangspunkt für die Generierung einer Legfolge ist dabei immer ein Leg l und eine Flotte f. l ist das erste Leg der zu generierenden Legfolge F und f ist die neue Flotte, die der Legfolge zugewiesen werden soll. In einer eingeschränkten Tiefensuche wird F schrittweise um *passende* Legs erweitert, bis eine Legfolge gefunden wird, die die Anforderungen an einen Change- bzw. Swap-Übergang erfüllt.

Auch hierzu finden sich die Details im Abschnitt 4.4.1.5.

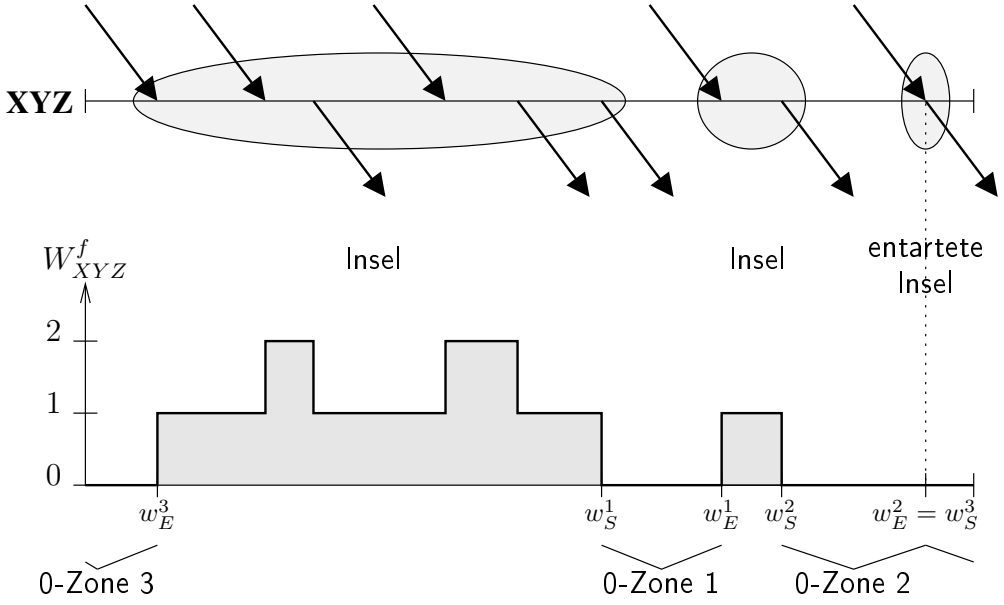


Abbildung 4.4: Wartefunktion, 0-Zonen und Inseln

4.3.2 Hill Climbing Heuristik

Der Hill Climbing Algorithmus ist einer der einfachsten Lokale Suche Algorithmen. Algorithmus 1 zeigt seine prinzipielle Vorgehensweise. Ausgehend von einer initialen Lösung durchsucht der Algorithmus die Nachbarschaft der aktuellen Lösung nach einer mit besserem Gewinnwert. Findet er eine solche, so wird diese zur neuen aktuellen Lösung, ansonsten wird der Algorithmus beendet.

Algorithmus 1 Hill Climbing Algorithmus

1: $S =$ initiale Lösung
2: **while** es gibt Nachbarn S_{new} von S mit $P(S_{new}) > P(S)$ **do**
3: $\quad S = S_{new}$
4: **return** S

Offensichtlich ist die von einem Hill Climbing Algorithmus generierte Lösung ein lokales Optimum. Die Folge der aktuellen Lösungen hat einen monoton steigenden Gewinnwert und endet in einem lokalen Maximum (oder „Gipfel“), daher der Name Hill Climber.

Es gibt zwei Varianten des Hill Climbing Algorithmus, die sich darin unterscheiden, welche bessere Lösung aus der Nachbarschaft der aktuellen Lösung ausgewählt wird:

- die erste im Verlauf der Suche gefundenen bessere Nachbarlösung
- die Nachbarlösung mit dem besten Gewinnwert

Da die Nachbarschaft einer Lösung zumeist sehr groß ist, mithin die komplette Durchsuchung lange dauert, wird meistens die erste Variante bevorzugt, da sie die geringere Laufzeit

verspricht. Auch unser Hill Climbing Algorithmus für das Flottenzuweisungsproblem arbeitet so. Dabei durchsucht er die Nachbarschaft der aktuellen Lösung zwar systematisch, aber in zufälliger Reihenfolge (siehe Abschnitt 4.4.1.5).

Der große Vorteil des Hill Climbing Algorithmus liegt in seiner geringen Laufzeit, vor allem im Vergleich zu Simulated Annealing oder Tabu Search Ansätzen, da zumeist nach vergleichsweise wenigen Schritten ein lokales Maximum gefunden wird. Allerdings kann die Lösungsqualität einer lokal-optimalen Lösung beliebig schlecht sein und der Hill Climbing Algorithmus ist damit sehr abhängig von der verwendeten initialen Lösung.

4.3.3 Simulated Annealing Heuristik

Die unerwünschte Eigenschaft des Hill Climbing Algorithmus, in einem schlechten lokalen Optimum enden zu können, hat zu einer ganzen Reihe von Metaheuristiken (Tabu Search, genetische Algorithmen, . . .) geführt, die diesen Makel zu beheben versuchen. Um aus lokalen Optima entkommen zu können, lassen diese Verfahren auch Übergänge zu Nachbarn zu, die einen schlechteren Gewinnwert besitzen.

Algorithmus 2 Simulated Annealing Algorithmus

```
1: S = initiale Lösung
2: T = Starttemperatur
3: repeat
4:    repeat
5:       S_new = Nachbarlösung von S
6:       Δ = P(S_new) - P(S)
7:       if (Δ > 0 oder r < e^(Δ/T)) then
8:          S = S_new
9:    until Gleichgewichtszustand erreicht
10:   Reduziere Temperatur T
11: until keine Verbesserungen mehr erreichbar
12: return S
```

Eine dieser Metaheuristiken ist Simulated Annealing; es handelt sich dabei eher um eine ganze Klasse von Algorithmen, die dem allgemeinen Schema von Algorithmus 2 folgen. Sie hat sich einerseits in der Praxis bei einer großen Anzahl von kombinatorischen Optimierungsproblemen als Lösungsheuristik bewährt, und ermöglicht andererseits durch ihre stochastischen Eigenschaften theoretische Aussagen über ihr Konvergenzverhalten [Hájek, 1985, Hájek, 1988].

Im Gegensatz zum Hill-Climber werden mit einer gewissen Wahrscheinlichkeit auch schlechtere Nachbarn akzeptiert. Die Wahrscheinlichkeit, mit der dieses passiert, hängt dabei zum einen vom Maß der möglichen Verschlechterung und zum anderen von einem Steuerungsparameter, genannt *Temperatur*, ab. Je größer die Verschlechterung und je kleiner die Temperatur, desto geringer ist die Wahrscheinlichkeit, dass ein verschlechternder Nachbarschaftsübergang akzeptiert wird.

Der Algorithmus iteriert nun dieses Generieren, Akzeptieren bzw. Verwerfen von Übergängen und senkt dabei im Verlauf die Temperatur immer weiter ab. Dies hat zur Folge, dass zum

Ende des Algorithmus fast nur noch verbessernde Übergänge akzeptiert werden. Schließlich wird die letzte im Verlauf des Algorithmus gefundene Lösung als Ergebnis zurückgegeben.

Die verschiedenen Simulated Annealing Varianten unterscheiden sich ihrem *Abkühlungsschema*: der Art und Weise, wie die Temperatur festgesetzt und verändert wird und wann die einzelnen Schleifen des Simulated Annealing Algorithmus beendet werden.

In unserer Implementierung verwenden wir ein adaptives Abkühlungsschema, das für Vehicle Routing Problem in [Osman, 1993] entwickelt worden ist. Dabei wird sowohl die Starttemperatur als auch die Abkühlungsgeschwindigkeit derart gewählt, dass sehr gute Lösungen in akzeptabler Zeit gefunden werden.

4.4 Details zur Swap-Change-Nachbarschaft

Hier beschreiben wir die Details zu der in Abschnitt 4.3 eingeführten Nachbarschaft, die von den Lokale Suche Verfahren zur Flottenzuweisung verwendet wird. Abschnitt 4.4.1 kümmert sich ausführlich um die Nachbarschaftsübergänge, wie sie generiert werden und welche Eigenschaften sie haben. Da durch die verwendete Nachbarschaft die Gesamtflugzeuganzahl einer Lösung kaum gezielt beeinflusst werden kann, widmet sich Abschnitt 4.4.2 Methoden, die versuchen, Lösungen mit weniger Flugzeugen und unterschiedlicher Flottenauslastung zu finden.

4.4.1 Nachbarschaft

Die Nachbarn einer aktuellen Lösung lassen sich in den in dieser Arbeit beschriebenen Heuristiken durch die beiden in Abschnitt 4.3.1 beschriebenen Übergänge Change und Swap erreichen; diese beiden Arten von Übergängen erhalten dabei die Balanciertheit der Lösung.

Sowohl Swaps als auch Changes bedienen sich Legfolgen, deren Flottenzuweisung geändert wird. Diese Legfolgen werden mittels einer eingeschränkten Tiefensuche generiert. Dies geschieht dynamisch während der Suche nach einem Nachbarn im Lokale Suche Optimierer.

Generiert man diese Legfolgen mehr oder weniger zufällig, hat das zur Folge, dass die Nachbarschaftsübergänge zwar die Balanciertheit erhalten, die Flugzeuganzahl aber meistens überschritten wird; man findet mit zufälligen Legfolgen nur sehr selten Übergänge, die zu einer *gültigen* Nachbarlösung führen, einer Lösung, die auch die vorgegebene Flugzeuganzahl einhält.

Da aber das Generieren einer Legfolge mittels Tiefensuche ein recht zeitaufwendiger Arbeitsschritt ist und ein Lokale Suche Algorithmus eine große Anzahl an Nachbarschaftsübergängen untersuchen muss, um gute Lösungen zu finden, sollte man im Hinblick auf die Laufzeit des Algorithmus am besten nur solche Übergänge erzeugen, die gewährleisten, dass die Flugzeuganzahl nicht verletzt wird.

Die Heuristiken verfolgen diesen Weg, indem sie zielgerichtet nur solche Übergänge erzeugen (bzw. überhaupt erzeugen können), die die Flugzeuganzahl jeder Flotte der aktuellen Lösung

nicht erhöht; verletzt die aktuelle Lösung die vorgegebene Flugzeuganzahl nicht, so gilt dies auch nach einem Nachbarschaftsübergang. Die Suche nach Übergängen, insbesondere das Generieren von passenden Legfolgen mittels Tiefensuche, wird durch diese Vorgabe kaum erschwert sondern, im Gegenteil, beschleunigt, da sich aus der Vorgabe, keine Flugzeuge zusätzlich benutzen zu wollen, Beschränkungen für die maximale Dauer von Legfolgen und für mögliche Anschluss-Legs ableiten lassen, die den Suchbaum der Tiefensuche einschränken.

Allerdings ergibt sich hieraus auch eine unter Umständen unerwünschte Eigenschaft der Nachbarschaft: Benutzt eine aktuelle Lösung weniger Flugzeuge als vorgegeben, so tun dies auch alle davon abgeleiteten Lösungen. Ein einmal eingespartes Flugzeug wird (ohne zusätzliche Maßnahmen) nie wieder reaktiviert, auch wenn sich dadurch höhere Gewinne erzielen ließen. In den meisten Fällen stellt diese Eigenschaft allerdings keine wirkliche Einschränkung dar, da eingesparte Flugzeuge die Kosten senken und gerne als Reserve für kurzfristige Störungen im Flugplan bereitgehalten werden.

Abschnitt 4.4.1.1 beschreibt, wie man Legfolgen grundsätzlich konstruieren muss, damit sie bei der Verwendung durch Swaps und Changes nicht zu einem Mehrverbrauch an Flugzeugen führen. Abschnitt 4.4.1.2 legt dar, welche speziellen zusätzlichen Anforderungen ein Change an die verwendete balancierte Legfolge stellt und Abschnitt 4.4.1.3 tut dies für die bei einem Swap involvierten Legfolgen. Abschnitt 4.4.1.4 beschreibt wie die Kostenänderung, die über das Akzeptieren bzw. Verwerfen eines Übergangs im SA-Algorithmus entscheidet, bestimmt wird, und der letzte Abschnitt 4.4.1.5 geht auf die zufällige Auswahl von Nachbarschaftsübergängen für die aktuelle Lösung ein.

4.4.1.1 Legfolgen

Die für die beiden Nachbarschaftsübergangsarten Change und Swap benötigten Legfolgen werden mit einer eingeschränkten Tiefensuche erzeugt. Dabei wird darauf geachtet, dass die so konstruierten Legfolgen nicht zu einem Flugzeugmehrverbrauch *auf den Zwischenstationen* führen, wenn ihnen eine neue Flotte zugewiesen wird.

Eingeschränkte Tiefensuche Sowohl Changes als auch Swaps beruhen darauf, Legfolgen eine geänderte Flotte zuzuweisen. In beiden Fällen werden diese Legfolgen nach einem gemeinsamen Prinzip, der eingeschränkten Tiefensuche, konstruiert. Die Tiefensuche unterliegt dabei einer Tiefenbeschränkung $t_{\max}$ und einer Gradbeschränkung $g_{\max}$.

Als Eingabedaten bekommt die Tiefensuche ein Leg l, dem eine Flotte $e = z(l)$ (alte Flotte) zugewiesen ist, und eine Flotte $f \neq e$ vorgegeben; l soll das erste Leg in der zu konstruierenden Legfolge sein und f stellt die neue Flotte dar, die an die Legfolge zugewiesen werden soll.

Allgemein muss die konstruierte Legfolge dabei gewährleisten, dass durch die Zuweisung der neuen Flotte an sie *keine zusätzlichen Flugzeuge auf den Zwischenstationen* benötigt werden, sowohl für die alte als auch für die neue Flotte. Weitere spezielle Anforderungen ergeben sich aus der Art des Übergangs (Change oder Swap) und werden in den entsprechenden Abschnitten weiter unten beschrieben.

Die Tiefensuche arbeitet rekursiv und bekommt in jedem Schritt eine Legfolge F übergeben. Initial besteht diese Legfolge dabei nur aus dem vorgegebenen Leg l. Während jedem Rekursionsschritt wird die Legfolge um ein Leg erweitert. Die Vorgehensweise stellt sich wie folgt dar:

- Erfüllt die Legfolge F die (von dem Übergang) an sie gestellten Bedingungen, merke dir diese Legfolge und beende diesen Schritt.
- Besteht die Legfolge aus $t_{\max}$ Legs, so ist die maximale Suchtiefe erreicht; breche diesen Rekursionsschritt dann ab.
- Ansonsten bestimme das Kandidaten-Intervall $[k_S, k_E] \subseteq [\mathcal{T}]$, während dem alle möglichen Folgelegs starten müssen, um keinen Mehrverbrauch an Flugzeugen zu verursachen.
- Bestimme die Menge der Kandidaten: alle Legs, die aktuell von der alten Flotte e geflogen werden, während des Kandidaten-Intervalls $[k_S, k_E]$ vom Ziel-Flughafen s^a_F der Legfolge F starten und auch von der neuen Flotte f geflogen werden können.
- Wähle aus der Menge der Kandidaten maximal $g_{\max}$ Legs aus, hänge an die Legfolge jeweils eines dieser Legs an und führe rekursiv einen Schritt mit der erweiterten Legfolge durch.

Zwei Punkte der Tiefensuche bedürfen noch weiterer Erklärungen: „Wie bestimmt man das Kandidaten-Intervall $[k_S, k_E]$?“ und „Welche $g_{\max}$ Kandidaten wählt man aus, um die Legfolge zu erweitern?“. Diesen Punkten widmen sich die nächsten beiden Abschnitte.

Kandidaten-Intervall Bei der Tiefensuche spielt das Kandidaten-Intervall eine zentrale Rolle: es legt auf den Zwischenstationen das Zeitintervall fest, während dem mögliche Folgelegs starten müssen, damit die generierte Legfolge keine zusätzlichen Flugzeuge auf den Zwischenstationen benötigt. Zur Bestimmung des Kandidaten-Intervalls werden hier die Mechanismen aus Abschnitt 4.3.1.2, die Wartefunktionen und Inseln von Flughäfen, verwendet.

Zunächst betrachten wir die Auswirkungen, die das Ändern der Flotte einer Legfolge auf einen betroffenen Zwischenflughafen haben. Aus Sicht der alten Flotte e werden zwei Flugereignisse, eine Landung und ein Start, auf der Zwischenstation gelöscht, aus Sicht der neuen Flotte f kommen diese beiden Flugereignisse neu hinzu. Im Bereich zwischen den beiden Flugereignissen ändern sich dabei die Werte der Wartefunktion.

Je nach der zeitlichen Lage des Start- und Landeereignisses zueinander ergeben sich die in Abbildung 4.5 gezeigten Auswirkungen für die Wartefunktion der alten bzw. neuen Flotte. $F_{|F|}$ stellt das Leg dar, dessen Landeereignis betroffen ist, und k bezeichnet das Leg des Startereignisses. Dabei wird die Landezeit von $F_{|F|}$ als fest angenommen und ΔW^f_{XYZ} gibt die Änderungen für die Wartefunktion in Abhängigkeit von der Startzeit von k an. Dies entspricht der Situation während der Tiefensuche: Für das letzte, fest vorgegebene Leg der Legfolge $F_{|F|}$ sollen alle passenden Anschlusslegs k gesucht werden.

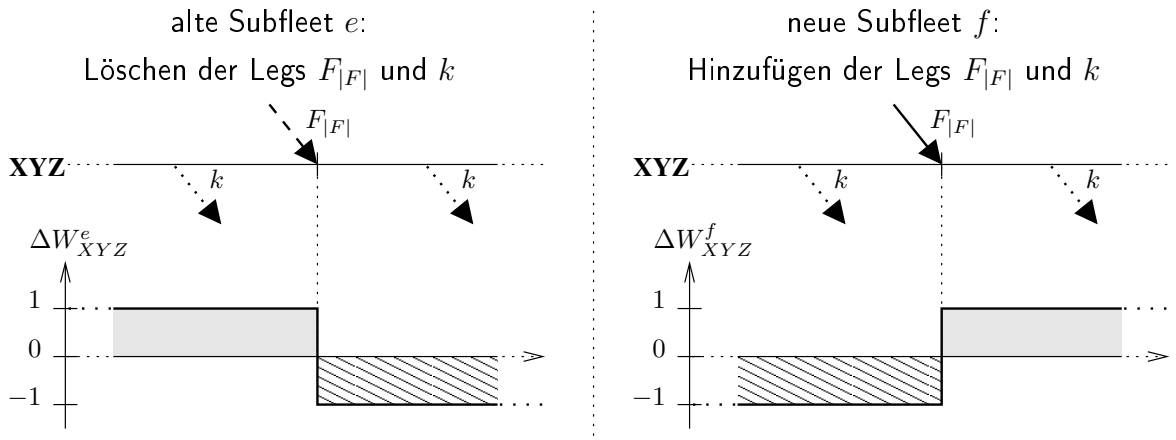


Abbildung 4.5: Auswirkungen auf die Wartefunktion von Zwischenstationen beim Ändern der Flottenzuweisung einer Legfolge

Für die alte Flotte nimmt der Wert der Wartefunktion zwischen Lande- und Startereignis um eins ab, falls k später startet als $F_{|F|}$ landet; ansonsten nimmt die Wartefunktion zwischen Start- und Landeereignis um eins zu. Für die neue Flotte f stellt sich die Situation genau umgekehrt dar.

Durch Änderungen an einer Wartefunktion werden nun genau dann zusätzliche Flugzeuge erforderlich, wenn ihr Wert wegen der Änderungen in einem Bereich negativ wird: Die Wartefunktion beschreibt die Anzahl der Flugzeuge einer Flotte, die auf einem Flughafen zur Verfügung stehen. Eine negative Anzahl ist dabei nicht zulässig und kann nur dadurch ausgeglichen werden, dass man zusätzliche Flugzeuge der betroffenen Flotte für die gesamte Periodendauer auf dem Flughafen bereitstellt.

Unter Zuhilfenahme dieser Vorüberlegungen lässt sich nun das Kandidaten-Intervall exakt bestimmen. Man sucht für ein ankommendes Leg $F_{|F|}$ das Zeitintervall, in dem alle möglichen Folgelegs starten müssen, um zu keinem Mehrverbrauch an Flugzeugen zu führen.

Alte Flotte Im Falle der alten Flotte werden das Leg $F_{|F|}$ und ein Folgeleg auf der Zwischenstation gelöscht, d.h. wenn das Folgeleg früher startet als $F_{|F|}$ landet werden die Werte der Wartefunktion in einem Bereich erhöht und wenn das Folgeleg später startet werden die Werte der Wartefunktion im Bereich zwischen Landung und Start um 1 verringert. Nur der letzte Fall ist interessant, da dadurch die Wartefunktion ggf. negativ werden kann.

Abbildung 4.6 zeigt das Leg $F_{|F|}$ auf der Zwischenstation, die Insel, zu der $F_{|F|}$ gehört, und die zugrundeliegende Wartefunktion. Die gestrichelte Linie beschreibt den Verlauf der Wartefunktion in Abhängigkeit vom Startzeitpunkt des Folgelegs, nachdem $F_{|F|}$ und das Folgeleg gelöscht worden sind.

Wie man sieht, werden die Werte der Wartefunktion für die alte Flotte negativ, sobald die Startzeit des Folgelegs größer ist als der Endzeitpunkt der Insel, zu der $F_{|F|}$ gehört. Das bedeutet, dass aus Sicht der alten Flotte alle Legs, die früher starten, mögliche Folgelegs

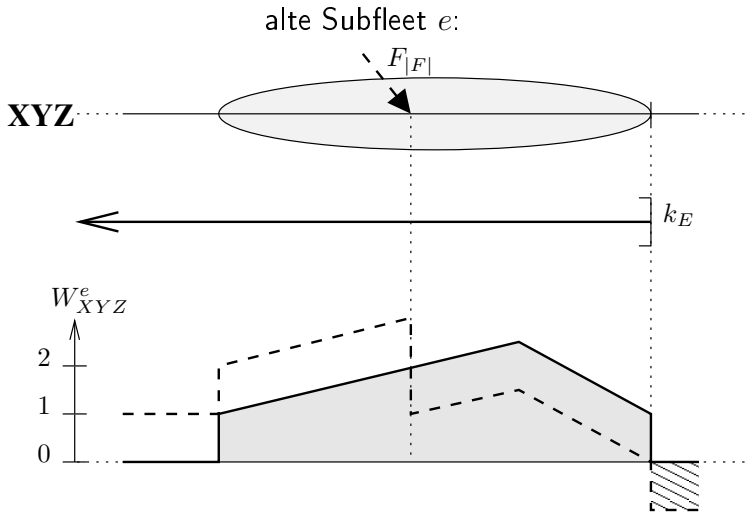


Abbildung 4.6: Rechte Schranke des Kandidaten-Intervalls

sein können. Die Endzeit der Insel von $F_{|F|}$ liefert also die rechte Schranke k_E des Kandidaten-Intervalls.

Neue Flotte Im Falle der neuen Flotte werden das Leg $F_{|F|}$ und ein Folgeleg auf der Zwischenstation hinzugefügt, d.h. wenn das Folgeleg später startet als $F_{|F|}$ landet werden die Werte der Wartefunktion in einem Bereich erhöht und wenn das Folgeleg früher startet werden die Werte der Wartefunktion im Bereich zwischen Start und Landung um 1 verringert. Nur der letzte Fall ist interessant, da dadurch die Wartefunktion ggf. negativ werden kann.

Abbildung 4.7 zeigt das Leg $F_{|F|}$ auf der Zwischenstation, zu der er hinzugefügt werden soll und die entsprechende Wartefunktion. Dabei sind zwei Fälle zu unterscheiden: links fällt der Landezeitpunkt von $F_{|F|}$ in einen Bereich, während dem kein Flugzeug verfügbar ist, rechts fällt der Landezeitpunkt in den Bereich einer existierenden Insel. Die gestrichelte Linie beschreibt jeweils den Verlauf der Wartefunktion in Abhängigkeit vom Startzeitpunkt des Folgelegs, nachdem $F_{|F|}$ und das Folgeleg hinzugefügt worden sind.

Sind zum Landezeitpunkt von $F_{|F|}$ keine Flugzeuge der neuen Flotte verfügbar (Abbildung 4.7 links), führt jedes Folgeleg mit einer kleineren Startzeit zu einem Mehrverbrauch von einem Flugzeug. In diesem Fall sind aus Sicht der neuen Flotte also nur solche Legs mögliche Folgeleg-Kandidaten, die später starten als $F_{|F|}$ landet. Die linke Schranke k_S des Kandidaten-Intervalls entspricht folglich der Landezeit von $F_{|F|}$.

Fällt der Landezeitpunkt von $F_{|F|}$ in den Bereich einer Insel (Abbildung 4.7 rechts), steht während der Insel ein Flugzeug der neuen Flotte zur Verfügung. Damit kann ein Folgeleg ohne Auswirkungen auf die Flugzeuganzahl während dieser Insel früher starten als $F_{|F|}$ landet, d.h. die linke Schranke k_S des Kandidaten-Intervalls fällt mit dem Beginn der Insel zusammen, in die $F_{|F|}$ fällt.

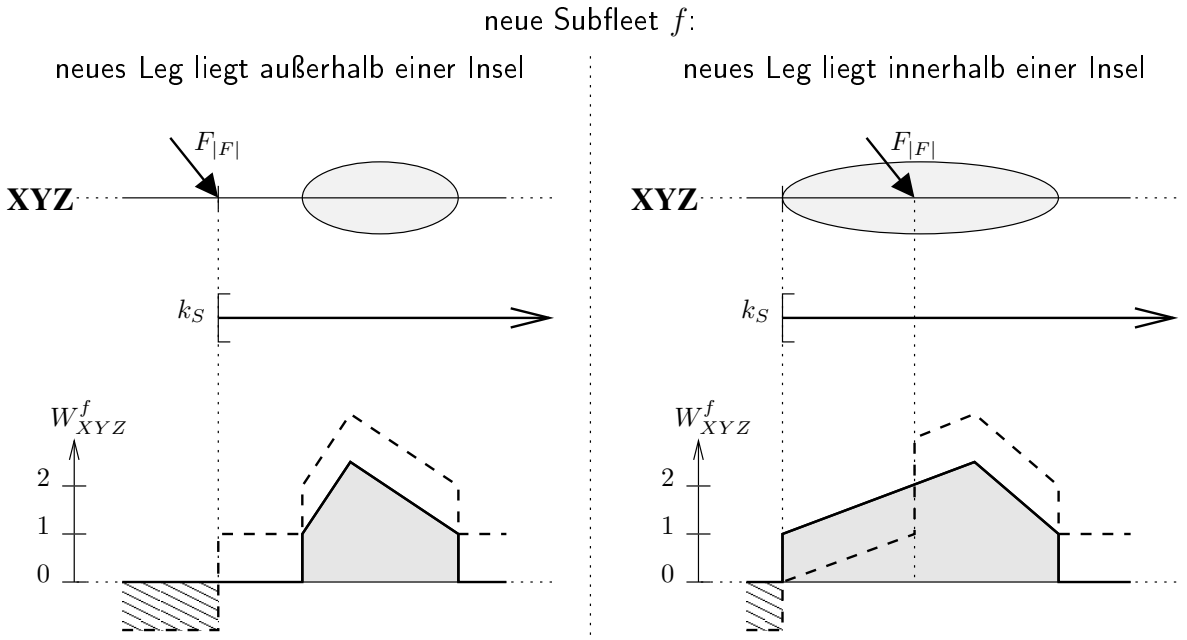


Abbildung 4.7: Linke Schranke des Kandidaten-Intervalls

Symbiose Aus der Situation der alten Flotte lässt sich eine rechte zeitliche Schranke k_E für das Kandidaten-Intervall ableiten, die Situation auf dem Zwischenflughafen der neuen Flotte liefert eine linke Schranke k_S. Diese beiden Schranken zusammen liefern also das Kandidaten-Intervall $[k_S, k_E]$ für das Leg $F_{|F|}$. Der linke Teil von Abbildung 4.8 stellt diesen Sachverhalt zusammenfassend dar.

Es bleibt anzumerken, dass die Landezeit von $F_{|F|}$ für die alte und neue Flotte im allgemeinen verschieden ist. In seltenen Fällen kann dadurch der Fall eintreten, dass das Kandidaten-Intervall leer ist (Abbildung 4.8 rechts); dann gibt es nur Folgelegs für $F_{|F|}$, die zu einem Mehrbedarf an Flugzeugen führen und es lassen sich keine Kandidaten finden. Die Legdauer von $F_{|F|}$ muss dabei für die alte Flotte kleiner sein als für die neue Flotte und die Insel der alten Flotte, zu der $F_{|F|}$ gehört, darf sich nicht viel länger über die Landezeit von $F_{|F|}$ hinaus

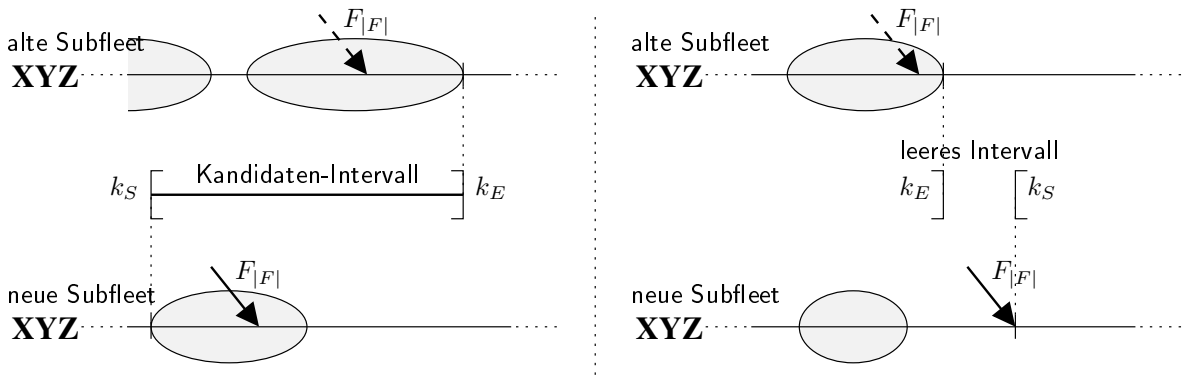


Abbildung 4.8: Kandidaten-Intervall

erstrecken.

Ist das Kandidaten-Intervall nicht leer, so gibt es auch mindestens ein Leg, das währenddessen vom Zwischenflughafen mit der alten Flotte startet, nämlich das letzte um k_E startende Leg, das zur Insel von $F_{|F|}$ gehört. Allerdings kann selbst dann die Menge der Kandidaten leer sein, da die neue Flotte die während des Kandidaten-Intervalls startenden Legs auch fliegen können muss.

Die mittels Kandidaten-Intervallen konstruierten Legfolgen weisen eine erwähnenswerte Besonderheit auf: sie lassen es auch zu, dass Folgelegs eher starten als ihre Vorgängerlegs landen. Diese Legfolgen können also nicht von einem Flugzeug nacheinander bedient werden. Dies ist nur dadurch ohne Flugzeugmehrverbrauch möglich, weil ein bereits auf der Zwischenstation wartendes Flugzeug die Legfolge mit dem Folgeleg fortführt, bevor das Flugzeug des Vorgängerlegs ankommt.

Kandidaten-Wahl Die Tiefensuche der Legfolgengenerierung unterliegt aus Gründen der Laufzeit einer Gradbeschränkung, d.h. die Legfolge wird in jedem Rekursionsschritt nur mit maximal $g_{\max}$ Kandidaten verlängert und weiter untersucht. Stehen mehr Kandidaten zur Verfügung, so werden nur $g_{\max}$ davon ausgewählt.

Alle Kandidaten sind als gleichwertig zu betrachten, was die Auswirkungen auf die Flugzeuganzahl betrifft, sie führen garantiert nicht zu einem Mehrverbrauch. Deshalb sind verschiedene Auswahlstrategien implementiert worden:

RANDOM Es werden rein zufällig $g_{\max}$ Legs aus der Kandidaten-Menge bestimmt.

TIME Es werden die $g_{\max}$ Legs ausgewählt, deren Startzeiten am nächsten an der Landezeit von $F_{|F|}$ liegen.

BEST Es werden die $g_{\max}$ Legs ausgewählt, die durch eine Flottenänderung von der alten Flotte e zur neuen Flotte f den profitabelsten Einfluss auf den Gewinn haben, d.h. für die $p_{l,e} - p_{l,f}$ am kleinsten ist.

4.4.1.2 Change

Beim Change-Übergang wird die Flotte einer *balancierten Legfolge* geändert. Als Vorgabe bekommt ein Change dabei ein Leg l und eine Flotte $f \neq z(l)$ geliefert. Ziel des Changes ist es, eine balancierte Legfolge mit l als erstem Leg zu generieren, der als neue Flotte f zugewiesen werden kann, ohne dabei mehr Flugzeuge als vorher zu benötigen.

Die Legfolge wird wie im vorherigen Abschnitt 4.4.1.1 beschrieben mittels eingeschränkter Tiefensuche erzeugt. Eine Bedingung, die ein Change dabei an die Legfolge stellt, ist ihre Balanciertheit, d.h. die Tiefensuche soll sich nur solche Legfolgen F merken, bei denen Start- und Zielflughafen gleich sind ($s_F^d = s_F^a$).

Mit allein dieser Bedingung liefert die Tiefensuche dann solche Legfolgen, die bei einem Change die Balanciertheit des Flugplans erhalten und die auf Zwischenstationen nicht zu

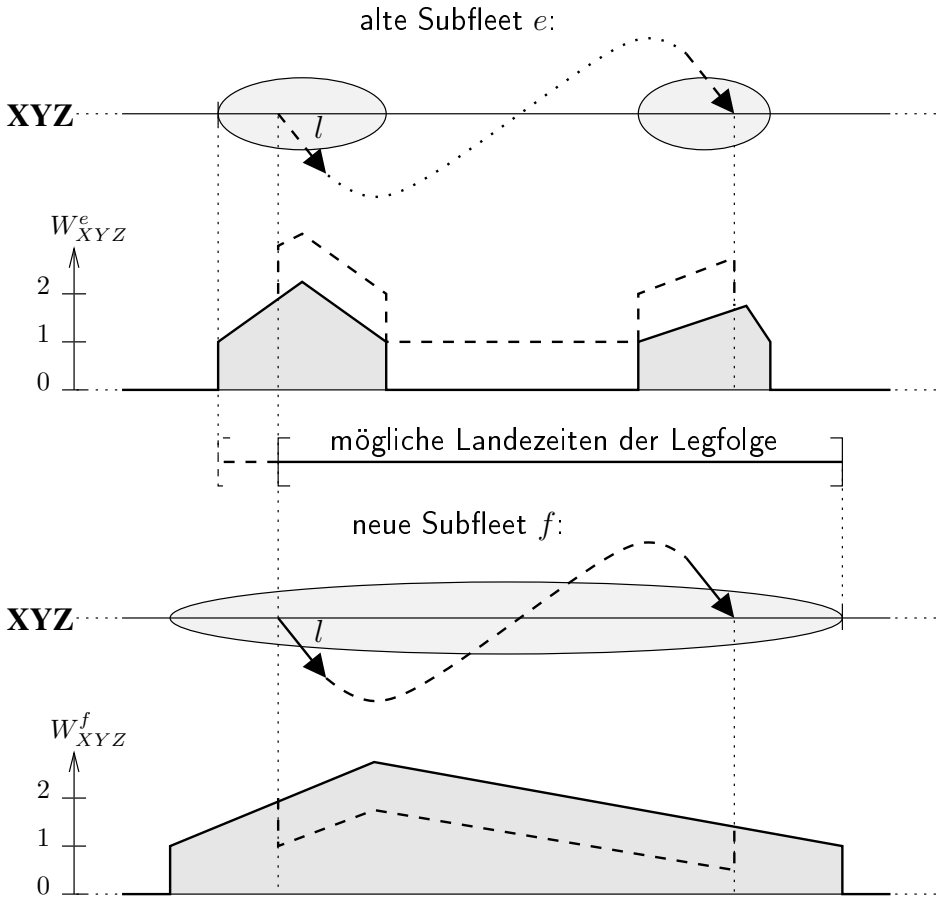


Abbildung 4.9: Change-Übergang

einem Mehrbedarf an Flugzeugen führen. Es muss also nur noch gewährleistet werden, dass auch auf dem Start-/Zielflughafen der Legfolge kein weiteres Flugzeug benötigt wird.

Die Situation bei einem Change stellt sich auf dem Start-/Zielflughafen wie in Abbildung 4.9 gezeigt dar. Eine balancierte Legfolge mit l als erstem Leg wird aus der alten Flotte gelöscht und zu der neuen Flotte f hinzugefügt. Die gestrichelte Linie stellt dabei jeweils den Verlauf der Wartefunktionen nach dem Change dar. Das Beispiel zeigt den normalen Fall, bei dem der Landezeitpunkt der Legfolge hinter dem Startzeitpunkt liegt. Hierbei steht nach dem Change während der Dauer der Legfolge ein zusätzliches Flugzeug der alten Flotte zur Verfügung und für die neue Flotte wird währenddessen ein Flugzeug benötigt.

Das bedeutet, dass in diesem Fall ein Mehrbedarf an Flugzeugen nur für die neue Flotte entstehen kann. Um dem entgegenzuwirken, muss während der gesamten Dauer der Legfolge auf dem Start-/Zielflughafen ein Flugzeug der neuen Flotte verfügbar sein, oder anders ausgedrückt: Die Startzeit der Legfolge (und damit die Startzeit des ersten Legs l) muss in den Bereich einer bestehenden Insel für die neuen Flotte f fallen und die Legfolge darf maximal bis zum Ende dieser Insel andauern. Diese Situation wird auch in Abbildung 4.9 dargestellt.

Damit ergeben sich die endgültigen Bedingungen, die ein Change an die Legfolgen stellt, die

von der Tiefensuche generiert werden:

- Die Legfolge muss balanciert sein.
- Die Legfolge darf höchstens so lange dauern, wie die Insel der neuen Flotte, in die die Startzeit des ersten Legs l fällt.

Steht zum Startzeitpunkt von l für die neuen Flotte kein Flugzeug zur Verfügung, wird die Tiefensuche erst gar nicht gestartet; es lässt sich keine Legfolge finden, die nicht zu einem Mehrbedarf an Flugzeugen führt. Ferner wird die Tiefensuche durch die Beschränkung der Legfolgendauer nochmals eingeschränkt: Es werden nur solche Kandidaten zur Verlängerung der Legfolge herangezogen, deren *Landezeit* innerhalb der maximalen Legfolgendauer liegt; im Endeffekt wird dadurch bereits die Dauer der Legfolgen begrenzt und gleichzeitig vermieden, Legfolgen, die eh schon zu lange dauern, weiter zu verlängern.

Bei allen obigen Überlegungen ist ein Aspekt der mittels Tiefensuche generierten Legfolgen unberücksichtigt geblieben: Auf Zwischenstationen können Folgelegs früher starten als ihre Vorgängerlegs landen. Dadurch ist es möglich, dass sich die Dauer einer Legfolge durch das Hinzufügen weiterer Legs insgesamt verkürzt. Dies kann soweit gehen, dass eine Legfolge früher landet als sie startet.

Standardmäßig werden solche Legfolgen für Changes nicht erzeugt, indem gefordert wird, dass die Landezeit aller Kandidaten größer sein muss als die Startzeit der Legfolge selbst. Optional kann diese Beschränkung allerdings aufgehoben werden, es ergibt sich dabei aber eine neue Zeitbarriere: Landet die Legfolge früher als sie startet, wird durch den Change im Bereich zwischen Landung und Start für die *alte* Flotte ein Flugzeug erforderlich,[4] d.h. der Anfangszeitpunkt der Insel der alten Flotte, zu der l gehört, ist der frühestmögliche Landezeitpunkt der Legfolge.

4.4.1.3 Swap

Der Swap-Übergang tauscht die Flotten der Legfolgen eines balancierten Legfolgenpaars untereinander aus. Wie der Change-Übergang bekommt er dabei als Vorgabe ein Leg l und eine Flotte $f \neq z(l)$ übergeben. Ziel ist es dabei, zu einer zu generierenden Legfolge mit l als erstem Leg eine passende Legfolge, die von Flotte f bedient wird, zu finden, so dass beide Legfolgen ein balanciertes Legfolgenpaar bilden und sich beim Austausch ihrer beiden Flotten kein Flugzeugmehrbedarf einstellt.

Ein Swap benötigt also zwei Legfolgen. Sie werden nacheinander mittels eingeschränkter Tiefensuche erzeugt. Zuerst wird nach einer Legfolge E mit l als erstem Leg gesucht, an die Flotte f zugewiesen werden kann. An die Legfolge E werden keine besonderen Bedingungen gestellt, außer der, dass sie möglichst aus $t_{\max}$ Legs bestehen soll.

Die zweite zu generierende Legfolge F, die aktuell von Flotte f geflogen wird und an die Flotte $e = z(l)$ zugewiesen werden soll, muss nun mit E bzw. einem Teil von E ein balanciertes Legfolgenpaar bilden. Abbildung 4.10 zeigt oben die Legfolge E und unten einen

[4] Bei der neuen Flotte kommt währenddessen ein zusätzliches Flugzeug hinzu.

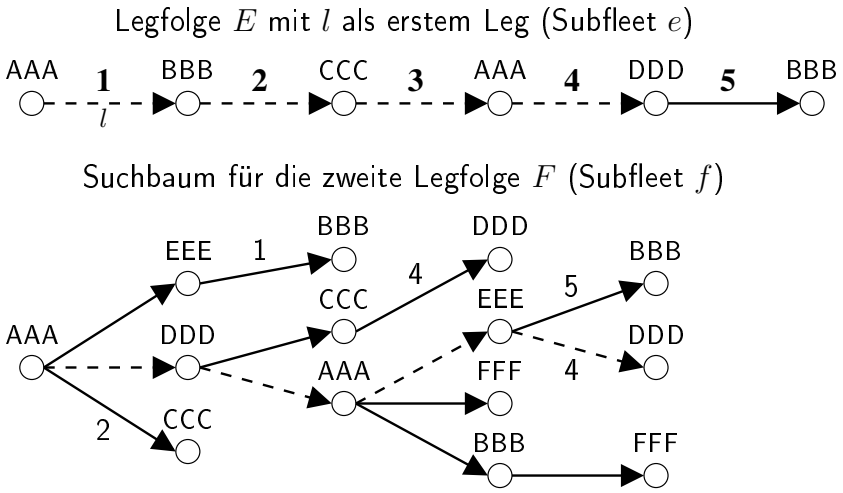


Abbildung 4.10: Swap-Übergang

Suchbaum, der bei der Tiefensuche nach F auftreten kann. Die gestrichelten Legs bilden dabei ein mögliches balanciertes Legfolgenpaar; dabei werden nur die ersten 4 Legs von E benutzt. Als Mindestvoraussetzung muss F folglich auf dem Startflughafen von E (und damit von l) starten und auf einem der Zielflughäfen der Legs aus E landen.

Obige Vorgehensweise garantiert, dass die Balanciertheit des Flugplans erhalten bleibt. Damit ferner die Flugzeuganzahl durch den Swap nicht erhöht wird, müssen die Start- und Landezeiten der beiden Legfolgen aneinander angepasst sein.[5] Ähnlich wie beim Kandidaten-Intervall der Tiefensuche lassen sich aus der Startzeit von E bzw. den Landezeiten der Legs von E Intervalle ableiten, in denen die Start- bzw. Ziellegs von F liegen müssen, damit das balancierte Legfolgenpaar beim Swap keine zusätzlichen Flugzeuge erforderlich macht. Wie man diese Intervalle genau bestimmt, ist weiter unten beschrieben.

Die Suche nach der zweiten Legfolge F verläuft nun wie folgt:

- Für jedes Leg k der Legfolge E wird das Intervall I_k bestimmt, während dem eine potentielle zweite Legfolge F landen muss, um keinen Flugzeugmehrverbrauch zu verursachen. Alle Legs, die während des Intervalls I_k auf dem Zielflughafen von k landen und die mit Flotte f geflogen werden, werden als mögliche Ziellegs der zweiten Legfolge F mit k markiert.
- Mittels der Startzeit von E wird das Intervall I bestimmt, während dem F starten muss. Mit allen Legs, die während des Intervalls I vom Startflughafen von F starten und die mit Flotte f geflogen werden, wird eine Tiefensuche gestartet. Dabei wird an die zu generierenden Legfolgen die Bedingung gestellt, dass sie mit einem im ersten Punkt markierten Zielleg enden müssen.

In Abbildung 4.10 sind die Legs von E mit den Zahlen von 1 bis 5 bezeichnet. Im Suchbaum von F finden sich diese Bezeichner an den markierten Ziellegs. Alle Pfade von der Wurzel zu

[5] Die Tiefensuche garantiert, dass auf den Zwischenstationen der beiden Legfolgen kein Mehrbedarf an Flugzeugen entsteht.

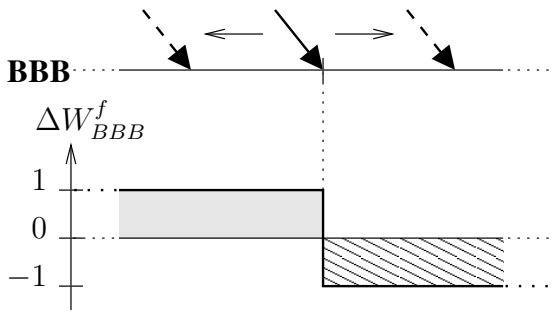


Abbildung 4.11: Verschieben eines Landeereignisses

einem dieser markierten Ziellegs stellen mögliche Legfolgen dar, die zusammen mit E bzw. einem Prefix davon ein balanciertes Legfolgenpaar ohne Flugzeugmehrverbrauch bilden.

Bevor die Intervalle, in denen die Start- bzw. Ziellegs liegen müssen, genauer beschrieben werden, folgen noch zwei Anmerkungen zur effizienten Implementierung von Swaps:

- Die Menge möglicher Startlegs von F ergibt sich aus der Startzeit der Legfolge E und damit des Legs l, sie lässt sich also schon vor der Generierung von E bestimmen. Ist diese Menge leer, kann kein Swap gefunden werden und E braucht erst gar nicht erzeugt zu werden.

- Weiter oben wurde erwähnt, dass an die Legfolge E keine besonderen Bedingungen gestellt werden. Mit der Kenntnis, wie die Suche nach der zweiten Legfolge verläuft, sollte aber die Legfolge E zumindest ein Zielleg markieren, da sonst während der zweiten Tiefensuche keine gültigen Legfolgen gefunden werden können.

Ziellegs Bei der Suche nach möglichen Ziellegs der zweiten Legfolge eines Swaps stellt sich einem folgende Ausgangssituation dar: Für ein bekanntes Leg k der ersten Legfolge E sollen alle möglichen Legs bestimmt werden, die für die zweite Legfolge als letztes Leg (Zielleg) in Frage kommen; sie müssen aktuell von Flotte f geflogen werden, den selben Zielflughafen wie k besitzen und zeitlich so zu k passen, dass auf dem Zielflughafen weder für die Flotte e noch für f ein Mehrbedarf an Flugzeugen entsteht.

Bei einem Swap wird auf dem Zielflughafen für die Flotte e bzw. f das Landeereignis eines Legs gelöscht, und dafür kommt das Landeereignis eines anderen Legs hinzu. *Aus der Sicht des Flughafens* spielen aber nur die Zeitpunkte, zu denen Flugereignisse stattfinden, eine Rolle, die sie verursachenden Legs sind nicht relevant. Folglich können die Auswirkungen eines Swaps auf den Zielflughafen auch als zeitliches *Verschieben* eines Landeereignisses interpretiert werden.

Abbildung 4.11 zeigt die Konsequenzen, die sich aus dem Verschieben von Landeereignissen für die Wartefunktion des entsprechenden Flughafens ergeben. Kritisch in Bezug auf die Flugzeuganzahl sind demnach solche Verschiebungen, bei denen das Landeereignis später zu liegen kommt, da hierdurch der Wert der Wartefunktion in einem Bereich um 1 kleiner wird.

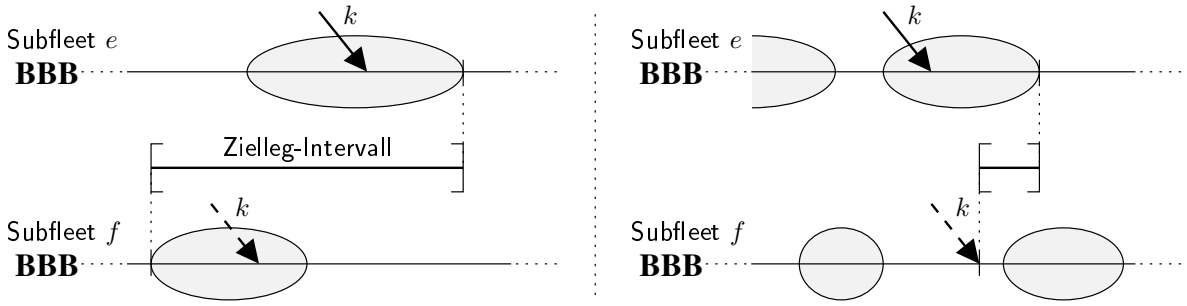


Abbildung 4.12: Intervall der Ziellegs eines Swaps

Daraus ergibt sich das Intervall, in dem die Landezeiten möglicher Ziellegs liegen müssen, wie in Abbildung 4.12 gezeigt. Die rechte Grenze wird dabei durch die Situation für die Flotte e vorgegeben. Hier wird das Landeereignis des Legs k zu dem Landeereignis eines Ziellegs hin „verschoben“; das Landeereignis des Ziellegs darf dabei nur dann später eintreten, wenn es innerhalb der Insel bleibt, zu der k gehört.

Die linke Grenze des Zielleg-Intervalls wird durch die Situation für Flotte f bestimmt. Hier wird das Landeereignis eines Ziellegs zum Landeereignis von k hin „verschoben“;[6] kritisch wird es hier also, wenn das Zielleg früher landet als Leg k. Dies ist nur erlaubt, wenn das Landeereignis von Leg k in eine bestehende Insel von Flotte f fällt, und auch dann nur bis zum Beginn dieser Insel (Abbildung 4.12 links). Ansonsten stellt der Landezeitpunkt von k selbst die linke Grenze des Intervalls dar (Abbildung 4.12 rechts).

Da die Zielleg-Intervalle auf den Landezeiten von Legs beruhen, die je nach Flotte unterschiedlich sein können, kann hier, ähnlich wie bei den Kandidaten-Intervallen aus Abschnitt 4.4.1.1, der Fall eintreten, dass ein leeres Intervall entsteht; dann gibt es für Leg k keine passenden Ziellegs. Aber selbst wenn das Zielleg-Intervall nicht leer ist, kann es sein, dass währenddessen auf dem Zielflughafen keine Legs mit Flotte f landen.

Startlegs Bei der Suche nach möglichen Startlegs der zweiten Legfolge eines Swaps stellt sich einem folgende Ausgangssituation dar: Für das vorgegebene erste Leg l der Legfolge E sollen alle möglichen Legs bestimmt werden, die für die zweite Legfolge als erstes Leg (Startleg) in Frage kommen; sie müssen aktuell von Flotte f geflogen werden, denselben Startflughafen wie l besitzen und zeitlich so zu l passen, dass auf dem Startflughafen weder für die Flotte e noch für f ein Mehrbedarf an Flugzeugen entsteht.

Bei einem Swap wird auf dem Startflughafen für die Flotte e bzw. f das Startereignis eines Legs gelöscht, und dafür kommt das Startereignis eines anderen Legs hinzu. Analog zu den Landeereignissen bei Ziellegs kann dies *aus der Sicht des Flughafens* als das zeitliche *Verschieben* eines Startereignisses angesehen werden. Wie aus Abbildung 4.13 ersichtlich ist, führt dabei ein Vorverlegen eines Startereignisses dazu, dass die Werte der Wartefunktion in dem Bereich der Verschiebung um 1 abnehmen. Vorverlegungen führen also zu einem Flugzeugmehrbedarf, wenn sie über die Insel, zu der das Startereignis gehört, hinaus durchgeführt werden.

[6] In diesem Fall ist also das Ziel der Verschiebung bekannt (k), nicht aber der Start (Zielleg).

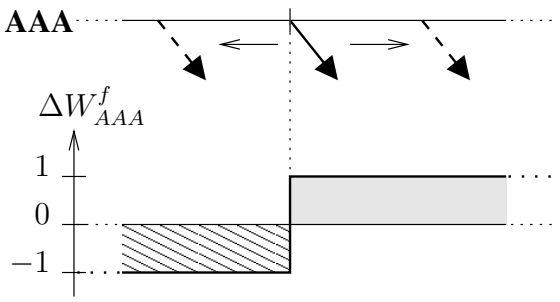


Abbildung 4.13: Verschieben eines Startereignisses

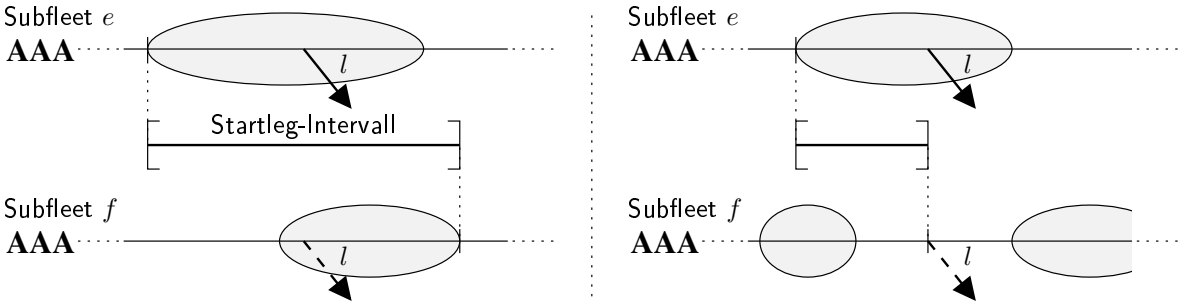


Abbildung 4.14: Intervall der Startlegs eines Swaps

Die Situation bei Startereignissen ist also genau entgegengesetzt zur Situation bei Landeereignissen. Folglich vertauschen bei der Bestimmung des Startleg-Intervalls die beiden Flotten ihre Rollen; Flotte e liefert die linke Grenze und Flotte f die rechte (Abbildung 4.14).

Der Beginn der Insel, zu der das Startleg l der ersten Legfolge gehört, liefert die linke Schranke des Startleg-Intervalls, da das Startereignis von l nicht darüber hinaus vorgezogen werden kann, ohne die Wartefunktion von Flotte e negativ zu machen. Leg l stellt für Flotte f das Ziel der Verschiebung des Startereignisses dar, Startlegs können also ohne Bedenken früher starten. Spätere Startzeiten sind nur erlaubt, wenn währenddessen auf dem Startflughafen ein Flugzeug von Flotte f verfügbar ist, die rechte Grenze des Startleg-Intervalls wird demnach durch den Startzeitpunkt von l oder, falls der Start von l in eine existierende Insel von Flotte f fällt, durch das Ende eben dieser Insel festgelegt.

Das Startleg-Intervall kann im Gegensatz zum Zielleg-Intervall nicht leer sein, da die Startzeiten aller Legs unverändert bleiben, egal mit welcher Flotte sie geflogen werden. Trotzdem ist es möglich, dass im Startleg-Intervall keine Legs vom Startflughafen mit Flotte f starten; es kann dann kein balanciertes Legfolgenpaar ohne Flugzeugmehrbedarf existieren, bei dem l das erste Leg der einen Legfolge ist und dessen andere Legfolge von Flotte f geflogen wird.

4.4.1.4 Auswirkung auf die Gewinnfunktion

Die Entscheidung, ob ein Nachbarschaftsübergang von einem Lokale Suche Algorithmus akzeptiert oder verworfen wird, wird anhand der Änderung ΔP der Gewinnfunktion, die

durch den Übergang bewirkt wird, getroffen. Der Wert von ΔP muss also für jeden Übergang bestimmt werden.

Sowohl ein Change als auch ein Swap weisen einer Menge von Legs $M \subset \mathcal{L}$ neue Flotten zu. Für ein Leg $l \in M$ sei $\hat{z}(l) \in \mathcal{F}$ diese neue Flotte. Da der Gewinn eines Legs allein von der das Leg fliegenden Flotte bestimmt wird, ergibt sich die Gewinnsdifferenz eines Changes bzw. Swaps einfach aus

$$\Delta P = \sum_{l \in M} \left(p_{l,\hat{z}(l)} - p_{l,z(l)} \right),$$

wobei es sich bei den einzelnen Gewinnen um vorgegebene konstante Werte handelt.

4.4.1.5 Wahl eines Nachbarn

In jedem Schritt eines Lokale Suche Algorithmus muss für die aktuelle Lösung ein zufälliger Nachbar erzeugt und bewertet werden. Im Falle der Swap-Change-Nachbarschaft muss man sich bei der Wahl eines Nachbarn zuerst auf die Art des Übergangs (Change oder Swap) festlegen. Sowohl bei einem Change $(l, f)_C$ als auch bei einem Swap $(l, f)_S$ müssen dann ein Leg l und eine neue Flotte f ausgewählt werden. l soll das erste Leg einer Legfolge sein, deren alte Flotte e gegen Flotte f ausgetauscht werden soll. Im Falle eines Changes ist die Legfolge balanciert und im Falle eines Swaps muss noch eine weitere passende Legfolge, geflogen von Flotte f, gesucht werden, der Flotte e zugewiesen wird.

Dabei kann es passieren, dass die Tiefensuche keine entsprechenden Legfolgen finden kann, die die Bedingungen erfüllen, die der Change bzw. Swap an sie stellen. In diesem Fall müssen solange andere Vorgaben $(l, f)_X$ ausgewürfelt werden, bis sich daraus ein gültiger Übergang generieren lässt.

Häufiger tritt jedoch der Fall ein, dass für eine Vorgabe $(l, f)_X$ während der Tiefensuche mehrere gültige Übergänge gefunden werden. Eine Vorgabe $(l, f)_X$ steht damit nicht für einen speziellen Nachbarschaftsübergang sondern für eine Gruppe von Übergängen mit ähnlichen Eigenschaften. Aus dieser Gruppe muss dann einer dieser Übergänge ausgewählt werden. In den Lokale Suche Heuristiken sind dazu verschiedene Varianten implementiert worden. Sie wählen folgenden Übergang aus:

- den ersten gefundenen Übergang (FIRST)
- den Übergang mit der besten Gewinndifferenz ΔP (BEST)
- den ersten Übergang mit positiver Gewinndifferenz; falls kein solcher existiert, den mit der besten Gewinndifferenz (ACCEPT)
- den ersten Übergang, dessen Gewinndifferenz oberhalb eines vorgegebenen Wertes liegt (BETTER)

FIRST Soll der erste gefundene Übergang ausgewählt werden, kann die Tiefensuche abgebrochen werden, sobald ein gültiger Übergang generiert ist. Diese Wahl stellt damit die schnellste Methode dar, einen Nachbarschaftsübergang zu finden.

Kombiniert man dies mit einer zufälligen Auswahl von Folgeleg-Kandidaten während der Tiefensuche (Abschnitt 4.4.1.1), werden rein zufällige Nachbarschaftsübergänge erzeugt.

BEST Dieses (wie auch die folgenden) Auswahlverfahren sucht im Rahmen der Vorgabe $(l, f)_X$ zielgerichtet nach „guten" Nachbarschaftsübergängen, die eher akzeptiert werden.

BEST wählt dabei den Übergang mit der besten Gewinndifferenz ΔP aus. Dafür muss während der Tiefensuche der Suchbaum vollständig abgearbeitet werden, diese Methode braucht für die Generierung von Übergängen also am längsten.

ACCEPT Ein Übergang wird auf jeden Fall dann akzeptiert, wenn seine Gewinndifferenz positiv ist. Sobald ein solcher Übergang gefunden ist bricht diese Methode die Tiefensuche ab, ansonsten arbeitet sie wie BEST. Dadurch wird zu einer Vorgabe $(l, f)_X$ genau dann ein akzeptierender Übergang gefunden, wenn dies auch die BEST-Auswahl tun würde.

Vor allem zu Beginn der Lokale Suche Algorithmen, wenn es noch viele Übergänge mit positiver Gewinndifferenz gibt, beschleunigt ACCEPT die Generierung von Übergängen gegenüber BEST.

BETTER Neben der Vorgabe $(l, f)_X$ wird hierbei während der Generierung eines Übergangs noch eine Schranke S vorgegeben, die die Gewinndifferenz eines Übergangs überschreiten muss. Sobald ein solcher Übergang gefunden ist, wird die Tiefensuche abgebrochen, ansonsten wird kein Übergang zurückgeliefert.

Diese Auswahlmethode ist in erster Linie für den Hill Climbing Algorithmus implementiert worden; hierbei wird S auf 0 gesetzt, da nur verbessernde Nachbarschaftsübergänge interessieren. Sie lässt sich aber auch für den Simulated Annealing Algorithmus verwenden: Man legt dabei *vor* der Generierung des Übergangs fest, welche Gewinndifferenz man noch akzeptieren will, und setzt dazu

$$S = T \cdot \ln(random) \quad \text{mit zufälligem } random \in [0, 1),$$

wobei T für die aktuelle Temperatur im Simulated Annealing Algorithmus steht. Wird ein entsprechender Übergang gefunden, so gilt für dessen Gewinndifferenz

$$\Delta P > S = T \cdot \ln(random) \quad \Longleftrightarrow \quad e^{\frac{\Delta P}{T}} > random,$$

die Schranke wird demnach gemäß Boltzmann-Verteilung festgelegt.

Wie bei ACCEPT wird hierbei genau dann ein akzeptierender Übergang gefunden, wenn auch BEST einen finden würde. Die Tiefensuche wird aber frühestmöglich abgebrochen.

Die sich aus den Vorgaben $(l, f)_X$ ergebene Nachbarschaft einer Lösung ist riesig. Dementsprechend groß ist auch die Anzahl an Iterationen, die ein Lokale Suche Algorithmus durchführen muss, um gute Lösungen zu finden. Eine schnelle Konvergenzrate ist im Hinblick auf die Gesamtlaufzeit vorteilhaft. Die zielgerichtete Suche nach akzeptierenden Übergängen,

wie sie von den Auswahlverfahren BEST, ACCEPT und BETTER betrieben wird, führt zu einer schnelleren Konvergenzrate im SA-Algorithmus.

Zusätzlich wird der Lösungsraum zwar zufällig, aber systematisch durchsucht. Dazu werden alle möglichen Vorgaben $(l, f)_X$ in einer Liste gespeichert, die zufällig permutiert wird. Die zur Generierung von Nachbarn nötigen Vorgaben werden nun nacheinander aus der Liste gewählt; kommt man dabei an das Ende der Liste, wird diese erneut zufällig permutiert und man beginnt von vorne.

4.4.2 Flugzeuganzahl

Die Flugzeuganzahl ist bei den in dieser Arbeit beschriebenen Lokale Suche Algorithmen ein kritischer Faktor. Durch beliebige Nachbarschaftsübergänge steigt die benötigte Flugzeuganzahl unkontrolliert stark an, sie lässt sich aber andererseits nur schwer verringern.

Allgemein macht die Beschränkung der Flugzeuganzahl das Fleet Assignment zu einem schweren Problem; ein besonderer Grund, warum Lokale Suche-Verfahren davon betroffen sind, liegt darin, dass das Verringern der Flugzeuganzahl im Verhältnis zur Nachbarschaft ein eher globales Ereignis darstellt, das sich kaum durch einzelne unkoordinierte Übergänge erreichen lässt.

Die Übergänge der Swap-Change-Nachbarschaft garantieren, dass durch sie keine zusätzlichen Flugzeuge erforderlich werden. Es gibt aber Situationen, in denen man die Flugzeuganzahl einer Lösung gezielt verringern oder den Flugzeugbedarf einer Flotte auf Kosten einer anderen Flotte verkleinern möchte:

- Überschreitet die Initiallösung eines Fleet Assignment Problems die vorgegebene Flugzeuganzahl, so muss diese verringert werden, um überhaupt zulässige Lösungen zu finden.
- Arbeitet man mit einer Kostenfunktion, die auch die fixen Kosten der tatsächlich eingesetzten Flugzeuge berücksichtigt, oder will man allgemein möglichst wenig Flugzeuge einsetzten, müssen auch Lösungen untersucht werden, die weniger Flugzeuge als vorgegeben benötigen und Flotten unterschiedlich stark auslasten.

Zu diesem Zweck sind in den Lokale Suche Algorithmen spezielle Verfahren implementiert worden, die versuchen, die Flugzeuganzahl einer Lösung insgesamt zu verringern oder ein von einer Flotte benötigtes Flugzeug in eine andere Flotte zu verschieben. Letzteres Verfahren dient dabei zum einen dazu, eine Lösung, die von einzelnen Flotten zu viele Flugzeuge benötigt und andere Flotten noch nicht vollständig auslastet, gültig zu machen. Zum anderen kann damit für Lösungen mit ungenutzten Flugzeugen die Auslastung der einzelnen Flotten geändert werden, beispielsweise können gezielt Flugzeuge kostenträchtiger Flotten eingespart werden.

Beide Verfahren können optional aktiviert werden, wobei sie periodisch, normalerweise nach jeder Temperaturstufe im SA-Algorithmus, aufgerufen werden. Sie sind damit nicht Teil der eigentlichen Nachbarschaft der Lokale Suche Heuristiken.

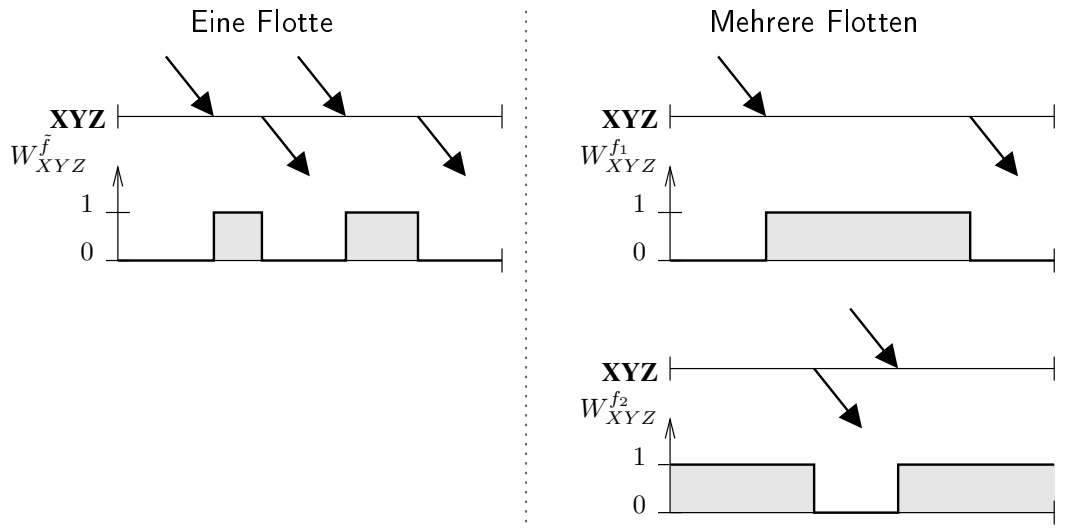


Abbildung 4.15: Eine Flotte vs. Mehrere Flotten

4.4.2.1 Verringern der Flugzeuganzahl

Das Verfahren zur Verringerung der Flugzeuganzahl stützt sich auf den in Abschnitt 4.7.1.1 erläuterten Vergleich von Lösungen mit einer und mehreren Flotten. Die Flugzeuganzahl, die der Flugplan benötigt, wenn er nur mit der einen virtuellen Flotte $\tilde{f}$ geflogen wird, stellt eine untere Schranke für die Flugzeuganzahl im Mehr-Flotten-Fall dar. Stimmt die Gesamtanzahl an Flugzeugen einer Lösung mit mehreren Flotten mit dieser unteren Schranke überein, kann die Flugzeuganzahl nicht weiter verringert werden.

Ansonsten besteht die Möglichkeit, dass die Flugzeuganzahl einer Lösung reduziert werden kann. Die im Vergleich zur virtuellen Flotte zusätzlich benötigten Flugzeuge werden dadurch verursacht, dass durch die Aufteilung auf verschiedene Flotten Legs die gemäß virtueller Flotte richtigen Anschlusslegs nicht erreichen können, da sie verschiedenen Flotten zugeteilt sind. Abbildung 4.15 zeigt ein einfaches Beispiel.

Flughäfen, auf denen so etwas passiert, lassen sich einfach ermitteln, wenn man die Summe der wartenden Flugzeuge aller Flotten zum Periodenende mit der Anzahl der wartenden Flugzeuge der virtuellen Flotte vergleicht. Ist die Summe größer, können auf diesem Flughafen möglicherweise Flugzeuge eingespart werden. Im Beispiel ist die Anzahl der wartenden Flugzeuge auf Flughafen XYZ zum Periodenende für die virtuelle Flotte 0, während die Summe für die zwei Flotten f_1 und f_2 1 ist.

Auf solchen Flughäfen wartet zu jedem Zeitpunkt der Standardperiode mindestens ein Flugzeug, allerdings nicht immer ein Flugzeug derselben Flotte. Die Wartezeit, die sich über die gesamte Standardperiode erstreckt, teilt sich auf verschiedene Flotten auf. Ziel ist es nun, diese Wartezeit bei einer Flotte zu kumulieren, d.h. von dieser Flotte wartet dann während der gesamten Standardperiode ein Flugzeug auf dem Flughafen, welches eingespart werden kann.

Anschaulich gesprochen müssen dazu *alle 0-Zonen einer Flotte* auf dem betroffenen Flughafen aufgefüllt werden. Ein Change bewirkt für die alte Flotte, dass die Werte der Wartefunktion

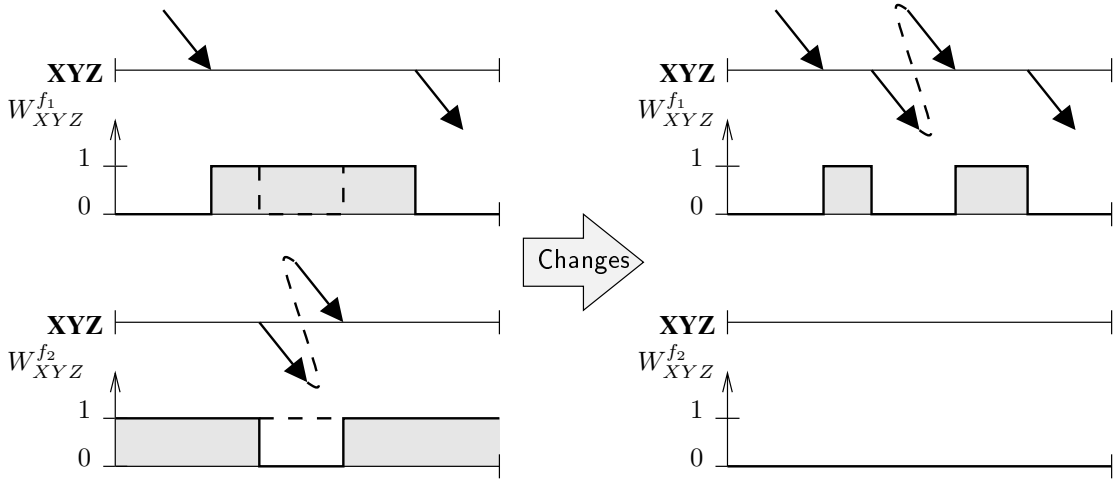


Abbildung 4.16: Einsparen eines Flugzeugs

zwischen Start und Landung des Changes um 1 größer werden. Überbrückt der Change dabei eine oder mehrere 0-Zonen der alten Flotte, werden diese aufgefüllt. Ein solcher Change lässt sich recht häufig finden, da während des Zeitraums der 0-Zonen irgendein Flugzeug einer anderen Flotte auf dem Flughafen zur Verfügung steht. Abbildung 4.16 zeigt, wie durch einen Change die einzige 0-Zone von Flotte f_2 aufgefüllt wird und dass dadurch die benötigte Flugzeuganzahl um 1 sinkt.

Das Verfahren zum Verringern der Flugzeuganzahl verläuft nun folgendermaßen:

- Es werden alle Flughäfen bestimmt, auf denen potentiell Flugzeuge eingespart werden können, indem die Anzahl der zum Periodenende wartenden Flugzeuge für die virtuelle Flotte mit der Summe der Flugzeuganzahlen der aktuellen Lösung verglichen wird.
- Auf jedem dieser Flughäfen wird nacheinander für jede Flotte versucht, ihre 0-Zonen durch Changes aufzufüllen; wenn dies gelingt, konnte ein Flugzeug eingespart werden und der Vorgang wird solange wiederholt, bis sich die Flugzeuganzahl nicht mehr verringern lässt.

Die Suche nach einem Change, der eine 0-Zone überbrückt, verläuft recht effizient; bei der anderen am Change beteiligten Flotte, der neuen Flotte, muss für die Dauer der 0-Zone ein Flugzeug zur Verfügung stehen, was die Wahl möglicher neuer Flotten einschränkt. Der Beginn der Insel der neuen Flotte bestimmt dabei, welche Legs der alten Flotte als Startleg für den Change in Frage kommen; ihre Startzeit muss innerhalb der Insel der neuen Flotte liegen. Für jedes dieser Legs wird mittels Tiefensuche nach einem Change gesucht, der die 0-Zone überbrückt.

4.4.2.2 Verschieben von Flugzeugen

Das Verschieben von Flugzeugen arbeitet so ähnlich wie das Verringern der Flugzeuganzahl. Für eine Flotte, die ein Flugzeug weniger benötigen soll, werden alle 0-Zonen eines Flughafens

durch Changes aufgefüllt. Dabei darf sich der Flugzeugbedarf einer anderen Flotte *einmal* um 1 erhöhen. Dadurch bleibt die Flugzeuganzahl insgesamt gleich.

Bei der Wahl des Flughafens, auf dem ein Flugzeug verschoben werden soll, werden solche Flughäfen bevorzugt, auf denen viele Flugereignisse stattfinden (das erleichtert die Suche nach Changes) und auf denen die Flotte nur wenige kurze 0-Zonen besitzt, die überspannt werden müssen.

Dadurch, dass sich während des Verschiebens die Flugzeuganzahl einer anderen Flotte erhöhen darf, wird in dieser anderen Flotte Platz geschaffen, um auch die übrigen Changes aufnehmen zu können, da die Erhöhung der Flugzeuganzahl ein Flugzeug dieser Flotte für die Dauer der gesamten Standardperiode verfügbar macht.

4.5 Verbindungsabhängige Mindestbodenzeiten und Gewinne

In diesem Abschnitt beschreiben wir, wie verbindungsabhängige Mindestbodenzeiten und Gewinne bei der Flottenzuweisung effizient berücksichtigt werden können. Wir stellen zum einen in Abschnitt 4.5.1 ein neues ganzzahliges lineares Modell vor, das für reale Instanzen die Vorteile des Time Space Network Modells (geringe Größe) mit denen des Connection Network Modells (hohe Ausdruckskraft) vereint. Zum anderen beschreiben wir in Abschnitt 4.5.2, wie die Lokale Suche Heuristiken modifiziert werden können, um verbindungsabhängige Mindestbodenzeiten zu unterstützen.

4.5.1 Hybrides Modell

Das jetzt vorgestellte ganzzahlige lineare Programm für das Flottenzuweisungsproblem ist eine Kombination aus Connection Network und Time Space Network Modell. Das Hybride Modell arbeitet nur dann korrekt, wenn alle verbindungsabhängigen Gewinne nicht-negativ sind. Dies ist allerdings keine Einschränkung, da mit modifizierten Gewinnen

$$\overline{p_{l,f}} = p_{l,f} + \min_{n \in A_{l,f}} \{p_{l,n,f}\}$$
$$\overline{p_{l,m,f}} = p_{l,m,f} - \min_{n \in A_{l,f}} \{p_{l,n,f}\}$$

sichergestellt werden kann, dass alle verbindungsabhängigen Gewinne größer gleich Null sind, ohne dass sich dabei die Lösungsbewertung ändert. Wir nehmen daher ohne Beschränkung der Allgemeinheit an, dass

- alle verbindungsabhängigen Gewinne nicht-negativ sind.

Damit das Hybride Modell in der Praxis gegenüber dem Connection Network Modell im Vorteil ist, das heißt lineare Programme mit (deutlich) weniger Variablen erzeugt, dürfen

- nur wenige, echt positive verbindungsabhängige Gewinne enthalten sein und
- sich die verbindungsabhängigen Mindestbodenzeiten $g_{l,m,f}$ eines jeden Leg-Flotten Paares (l, f) nicht allzusehr voneinander unterscheiden.

Beide Bedingungen werden von realen Probleminstanzen typischerweise erfüllt (siehe Abschnitt 2.2.1).

4.5.1.1 Zyklisches Hybrides Modell

Da das Hybride Modell unter anderem von einem Time Space Network Modell abgeleitet wird, überlegen wir uns zuerst, wie man für eine Flottenzuweisungsinstanz mit verbindungsabhängigen Mindestbodenzeit ein Time Space Network Modell konstruieren kann, dass nur *zulässige* Zuweisungen als Lösung besitzt, notwendigerweise aber *nicht alle* zulässigen Zuweisungen.

Dazu müssen wir zu jedem Leg-Flotten-Paar (l, f) eine flottenabhängige Mindestbodenzeit $g_{l,f}$ derart angeben, dass durch sie für eine zulässige Lösung im Time Space Network Modell garantiert wird, dass keine verbindungsabhängigen Mindestbodenzeiten verletzt werden:

$$g_{l,f} = \min \left\{ g^*_{l,m,x} \mid m \in A_{l,f} : \forall n \in A_{l,f} : g_{l,n,x} \leq g^*_{l,m,x} + (t^d_{n,f} \ominus_{\mathcal{T}} t^d_{m,f}) \right\}$$

Die Mindestbodenzeit $g_{l,f}$ wird auf die tatsächliche Bodenzeit $g^*_{l,m,f}$ gesetzt, die ein Flugzeug von Flotte f auf dem Flughafen verbringt, wenn es als nächstes das spezielle Leg m bedient. Dadurch entspricht der Ankunftszeitpunkt von Leg l im Time Space Network der Abflugzeit von Leg m. Daraus ergibt sich *im Time Space Network Modell* für ein beliebiges Leg n, das vom Zielflughafen von l startet, eine zusätzliche Bodenzeit von $t^d_{n,f} \ominus_{\mathcal{T}} t^d_{m,f}$, also eine tatsächliche Bodenzeit von $g^*_{l,m,f} + (t^d_{n,f} \ominus_{\mathcal{T}} t^d_{m,f})$. Das Leg m wird nun minimal so gewählt, dass dessen tatsächliche Bodenzeit gerade groß genug ist, damit die tatsächlichen Bodenzeiten im Time Space Network die Mindestbodenzeiten im Connection Network nicht unterschreiten. Die Minimumsbildung für $g_{l,f}$ ist wohl definiert, da zumindest das Leg n mit der größten Mindestbodenzeit $g_{l,n,f}$ eine zulässige Wahl darstellt.

Damit sind alle zulässigen Lösungen für das Time Space Network auch zulässig im Connection Network. Allerdings kann es passieren, dass im Time Space Network einzelne Verbindungen zwischen zwei Legs l und m mehr Bodenzeit verbrauchen (und damit unter Umständen mehr Flugzeuge), als dies tatsächlich notwendig wäre. Es können also nicht alle zulässigen Flottenzuweisungen von dem so konstruierten Time Space Network Modell erzeugt werden.

Abbildung 4.17 zeigt für ein Leg l die Wahl der flottenabhängigen Boden- und damit auch Ankunftszeit. Ab dem flottenabhängigen Ankunftszeitpunkt von Leg l stellen alle startenden Legs zulässige Folgelegs für l dar. In dem Intervall zwischen der minimalen verbindungsabhängigen Ankunftszeit von Leg l und seiner flottenabhängigen Ankunftszeit können sich so allerdings startende Legs befinden, die eigentlich zulässige Folgelegs sind aber im Time Space Network nicht erreicht werden können.

Im Hybriden Modell werden die fehlenden Verbindungsmöglichkeiten dadurch realisiert, dass dem Time Space Network Modell noch das Connection Network Modell zur Seite gestellt

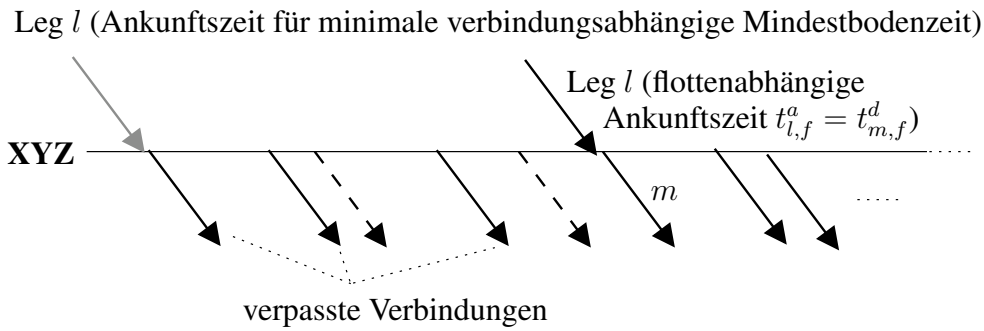


Abbildung 4.17: Wahl der flottenabhängigen Ankunftszeit im Time Space Network. Gestrichelte ausgehende Legs stellen unzulässige Folgelegs für Leg l dar.

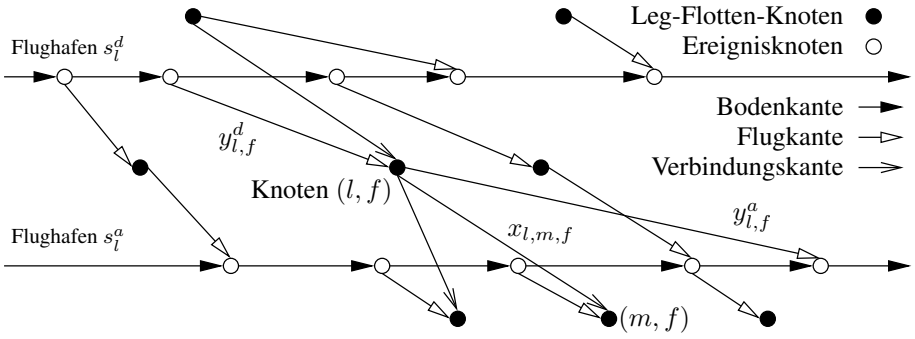


Abbildung 4.18: Ausschnitt aus einem Hybriden Modell

wird. Die Knoten des dem Hybriden Modell zugrunde liegenden Flussnetzwerks bestehen aus der Vereinigung der Knotenmengen des Time Space Networks und des Connection Networks. Die Kanten des Connection Networks werden unverändert übernommen. Gleiches gilt für die Bodenkanten des Time Space Networks. Die Flugkanten des Time Space Networks werden in zwei Teile aufgespalten: Die Flugkante für das Leg-Flotten-Paar (l, f) macht einen Zwischenstopp auf dem zugehörigen Leg-Flotten-Paar-Knoten des Connection Networks. Im Leg-Flotten-Paar-Knoten kann nun alternativ entschieden werden, ob die Reise eines Flugzeugs über die ausgehende Flugkante im Time Space Network weiter verlaufen soll oder über eine Verbindungskante im Connection Network. Abbildung 4.18 zeigt einen Ausschnitt aus einem Hybriden Netzwerk.

Das zyklische Hybride Modell kann damit wie folgt formuliert werden, wobei alle Bezeichner aus den Definitionen des Connection Network (Abschnitt 2.2.2) und Time Space Network Modells (Abschnitt 4.2) übernommen werden.

Modell 4.10 (Zyklisches Hybrides Modell).

$$\textit{Maximiere} \sum_{l \in \mathcal{L}} \sum_{f \in \mathcal{F}_l} p_{l,f} \left(y^a_{l,f} + \sum_{m \in A_{l,f}} p_{l,m,f} x_{l,m,f} \right) \tag{4.15}$$

unter den Nebenbedingungen

$$\sum_{f\in\mathcal{F}_l}\left(y^a_{l,f}+\sum_{m\in A_{l,f}}x_{l,m,f}\right)=1 \qquad \forall l\in\mathcal{L} \quad (4.16)$$

$$\sum_{(l,f)\in\mathcal{L}^a_v}y^a_{l,f}-\sum_{(l,f)\in\mathcal{L}^d_v}y^d_{l,f}+z_{v^-,v}-z_{v,v^+}=0 \qquad \forall v\in V \quad (4.17)$$

$$\sum_{k\in B_{l,f}}x_{k,l,f}-\sum_{m\in A_{l,f}}x_{l,m,f}+y^d_{l,f}-y^a_{l,f}=0 \qquad \forall l\in\mathcal{L}, f\in\mathcal{F}_l \quad (4.18)$$

$$\sum_{l\in\mathcal{L}:f\in\mathcal{F}_l}\left(\Delta_{l,f}y^a_{l,f}+\sum_{m\in A_{l,f}}\Delta_{l,m,f}x_{l,m,f}\right)+\sum_{v\in V^\Delta_f}z_{v^-,v}\le N_f \qquad \forall f\in\mathcal{F} \quad (4.19)$$

$$x_{l,m,f}\in\{0,1\} \qquad \forall l\in\mathcal{L}, f\in\mathcal{F}_l, m\in A_{l,f} \quad (4.20)$$

$$y^a_{l,f},y^d_{l,f}\in\{0,1\} \qquad \forall l\in\mathcal{L}, f\in\mathcal{F}_l \quad (4.21)$$

$$z_{v,v^+}\in\mathbb{N}_0 \qquad \forall v\in V \quad (4.22)$$

Die Zielfunktion (4.15) kombiniert die Zielfunktionen des Time Space und Connection Network Modells. Gleichungen (4.17) sind die Flusserhaltungsgleichungen des Time Space Networks. Da die Flugkanten aufgeteilt worden sind, unterscheiden wir hier jetzt zwischen abfliegenden ($y^d_{l,f}$) und ankommenden Flugkanten ($y^a_{l,f}$). Gleichungen (4.18) entsprechen den Flusserhaltungsgleichungen des Connection Networks. Neben den Verbindungskanten müssen hier jetzt auch die ankommenden und abfliegenden Flugkanten eines Leg-Flotten-Paares (l,f) berücksichtigt werden. Hierdurch werden das Time Space Network und das Connection Network miteinander gekoppelt. Ungleichungen (4.19) beschränken die Flugzeuganzahl und stellen eine Kombination der entsprechenden Ungleichungen im Time Space und Connection Network Modell dar. Flugzeuge können jetzt durch Flugkanten, Verbindungskanten oder Bodenkanten verbraucht werden. Die Variablen sind in (4.20) bis (4.22) wie für die ursprünglichen Modelle definiert.

Der Grund, warum negative verbindungsabhängige Gewinne $p_{l,m,f}$ in diesem Modell nicht erlaubt sind, ist, dass ansonsten nicht ausgeschlossen werden kann, dass ein Flugzeuge nacheinander die Legs l und m bedient, ohne die dafür anfallenden Kosten zu tragen, indem es über die Bodenkanten des Time Space Networks ausweicht. Dies ist zwar auch bei positiven verbindungsabhängigen Kosten möglich, allerdings erzwingt hier die Zielfunktion für optimale Lösungen, dass die gewinnbringendere Alternative über die Verbindungskante des Connection Networks gewählt wird.

Eine wichtige Frage verbleibt.

> Was ist überhaupt der Vorteil des Hybriden Modells gegenüber dem Connection Network Modell?

Beide Modelle sind gleich ausdrucksstark, aber in der bisher beschriebenen Form ist das Hybride Modell sogar größer als das Connection Network Modell - ein komplettes Time Space Network Modell ist dazugekommen.

Der springende Punkt ist, dass im Hybriden Modell nicht alle Verbindungskanten des Connection Networks notwendig sind, um alle zulässigen Lösungen erzeugen und bewerten zu können.

- Aus Sicht der Zielfunktion reichen all die Verbindungskanten, für die echt positive Gewinne definiert sind aus, da für Verbindungskanten mit Gewinn Null der alternative Weg durch das Time Space Network gleich viel Gewinn erbringt.
- Für die Zulässigkeit reicht es aus, nur die Verbindungskanten zu berücksichtigen, für die ansonsten das Time Space Network eine zu lange tatsächliche Bodenzeit verursachen würde.

Daraus folgt, dass die Mengen $A_{l,f}$ und $B_{l,f}$ wie folgt verkleinert werden können, ohne die Menge der zulässigen Flottenzuweisungen und deren Bewertung zu verändern:

$$A_{l,f} = \{m \in \mathcal{L} \mid s^d_m = s^a_l \wedge f \in \mathcal{F}_m \wedge (p_{l,m,f} > 0 \vee g^*_{l,m,f} < g_{l,f})\}$$
$$B_{l,f} = \{k \in \mathcal{L} \mid s^a_k = s^d_l \wedge f \in \mathcal{F}_k \wedge (p_{k,l,f} > 0 \vee g^*_{k,l,f} < g_{k,f})\}$$

Bei realen Instanzen mit wenigen echt positiven verbindungsabhängigen Gewinnen und nur verhältnismäßig kleinen Variationen in den verbindungsabhängigen Mindestbodenzeiten führt dies dazu, dass die Mengen $A_{l,f}$ und $B_{l,f}$ konstante Größe haben und die Variablenanzahl des Modells dadurch (wie im Time Space Network Modell) nur linear in der Leganzahl wächst. Im worst case ist das Wachstum aber wie im Connection Network Modell quadratisch in der Leganzahl.

4.5.1.2 Azyklisches Hybrides Modell

Der Vollständigkeit halber wollen wir jetzt noch kurz das azyklische Hybride Modell vorstellen. Die Grundidee ist die selbe wie im zyklischen Fall. Wir entfernen dazu aus den Mengen $\bar{A}_{l,f}$ und $\bar{B}_{l,f}$ die Hilfslegs $*$. Ihre Rolle wird vom Time Space Network übernommen. Ferner definieren wir die flottenabhängige Mindestbodenzeit $g_{l,f}$ für das Time Space Network Modell wie folgt:

$$g_{l,f} = \min\{g^*_{l,m,f} \mid m \in \bar{A}_{l,f} : \forall n \in \mathcal{L} : (s^d_n = s^a_l \wedge f \in \mathcal{F}_n \wedge t^d_{n,f} \geq t^d_{m,f}) \Rightarrow n \in \bar{A}_{l,f}\} \cup \{\infty\}$$

So ist sichergestellt, dass nach dem Ankunftszeitpunkt $t^a_{l,f}$ eines Legs l im Time Space Network nur noch Legs den Zielflughafen von l verlassen, die auch zu einer zulässigen Verknüpfung führen. Da im azyklischen Fall die Menge $\bar{A}_{l,f}$ leer sein kann, wird in diesem Fall die Mindestbodenzeit $g_{l,f}$ auf Unendlich gesetzt.

Damit lässt sich das azyklische Hybride Modell wie folgt aufstellen:

Modell 4.11 (Azyklisches Hybrides Modell).

$$\textit{Maximiere} \sum_{l \in \mathcal{L}} \sum_{f \in \mathcal{F}_l} p_{l,f} \left(y^a_{l,f} + \sum_{m \in \bar{A}_{l,f}} p_{l,m,f} x_{l,m,f} \right) \tag{4.23}$$

unter den Nebenbedingungen

$$\sum_{f\in\mathcal{F}_l}\left(y^a_{l,f}+\sum_{m\in\bar{A}_{l,f}}x_{l,m,f}\right)=1 \qquad \forall l\in\mathcal{L} \qquad (4.24)$$

$$\sum_{(l,f)\in\mathcal{L}^a_v}y^a_{l,f}-\sum_{(l,f)\in\mathcal{L}^d_v}y^d_{l,f}+z_{v^-,v}-z_{v,v^+}=0 \qquad \forall v\in V \qquad (4.25)$$

$$\sum_{k\in\bar{B}_{l,f}}x_{k,l,f}-\sum_{m\in\bar{A}_{l,f}}x_{l,m,f}+y^d_{l,f}-y^a_{l,f}=0 \qquad \forall l\in\mathcal{L},f\in\mathcal{F}_l \qquad (4.26)$$

$$\sum_{v\in V^\Delta_f}z_{*,v}\le N_f \qquad \forall f\in\mathcal{F} \qquad (4.27)$$

$$x_{l,m,f}\in\{0,1\} \qquad \forall l\in\mathcal{L},f\in\mathcal{F}_l,m\in\bar{A}_{l,f} \qquad (4.28)$$

$$y^a_{l,f},y^d_{l,f}\in\{0,1\} \qquad \forall l\in\mathcal{L},f\in\mathcal{F}_l \qquad (4.29)$$

$$z_{v,v^+}\in\mathbb{N}_0 \qquad \forall v\in V \qquad (4.30)$$

$$z_{*,v}\in\mathbb{N}_0 \qquad \forall v\in\bigcup_{f\in\mathcal{F}}V^\Delta_f \qquad (4.31)$$

Die Mengen $\bar{A}_{l,f}$ und $\bar{B}_{l,f}$ des azyklischen Connection Network Modells lassen sich dabei mit derselben Begründung wie im zyklischen Fall wie folgt verkleinern:

$$\bar{A}_{l,f}=\{m\in\mathcal{L}\mid s^d_m=s^a_l\wedge f\in\mathcal{F}_m\wedge t^a_{l,m,f}\le t^d_{m,f}\wedge(p_{l,m,f}>0\vee t^d_{m,f}<t^a_{l,f})\}$$
$$\bar{B}_{l,f}=\{k\in\mathcal{L}\mid s^a_k=s^d_l\wedge f\in\mathcal{F}_k\wedge t^a_{k,l,f}\le t^d_{l,f}\wedge(p_{k,l,f}>0\vee t^d_{l,f}<t^a_{k,f})\}$$

Eine Verbindungskante ist nur dann explizit erforderlich, wenn sie einen echt positiven Gewinn beisteuert oder eine Verknüpfung von Legs erlaubt, die im Time Space Netzwerk nicht möglich ist.

4.5.2 Erweiterung der Lokale Suche Heuristiken

Die jetzt vorgestellte Erweiterung der Lokale Suche Heuristiken erlaubt es, mit ihnen auch Probleminstanzen mit verbindungsabhängigen Mindestbodenzeiten zu optimieren. Verbindungsabhängige Gewinne werden allerdings nicht unterstützt. Wie beim Hybriden Modell sollten sich bei den Eingabeinstanzen für die erweiterten Heuristiken

- die verbindungsabhängigen Mindestbodenzeiten $g_{l,m,f}$ eines jeden Leg-Flotten Paares (l,f) nicht allzusehr voneinander unterscheiden.

Wir beschreiben hier wie in Abschnitt 4.3 nur die Erweiterungen für das zyklische Flottenzuweisungsproblem.

Die Grundidee der Erweiterung sieht folgendermaßen aus. Die eigentliche Generierung der Nachbarschaftsübergänge bleibt unverändert. Die benötigten Flugzeuganzahlen werden weiterhin mittels der Wartefunktionen ermittelt. Dazu müssen passende flottenabhängige Mindestbodenzeiten für jedes Leg-Flotten Paar definiert werden. Anders als im Hybriden Modell

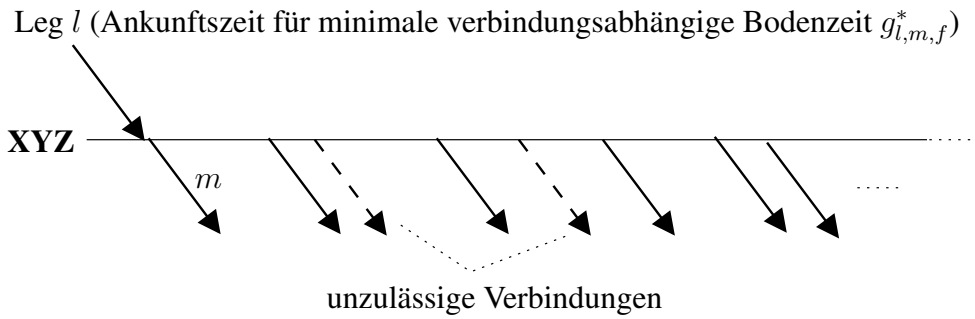


Abbildung 4.19: Wahl der flottenabhängigen Ankunftszeit für Lokale Suche Heuristiken. Gestrichelte ausgehende Legs stellen unzulässige Folgelegs für Leg l dar.

werden sie jedoch hier so gewählt, dass alle zulässigen Verknüpfungen prinzipiell erlaubt sind. Dadurch kann allerdings der Fall eintreten, dass auch unzulässige Nachbarn generiert werden. Daher wird vor der Entscheidung, ob während der Tiefensuche tatsächlich ein zulässiger Übergang gefunden worden ist, überprüft, ob der neue Nachbar auch alle verbindungsabhängigen Mindestbodenzeiten beachtet.

Die flottenabhängige Mindestbodenzeit eines Leg-Flotten Paares (l, f) wird für die Erweiterung der Heuristiken wie folgt definiert:

$$g_{l,f} = \min \left\{ g^*_{l,m,f} \mid m \in \mathcal{L} \wedge f \in \mathcal{F}_m \right\}$$

Damit wird, wie oben beschrieben, erreicht, dass nach dem Ankunftsereignis eines Legs l alle zulässigen Folgelegs später abfliegen und somit von Leg l erreicht werden können. Leider sind so allerdings potentiell auch Legs erreichbar, die *keine* zulässige Verknüpfung darstellen.[7]

Abbildung 4.19 zeigt für ein Leg l die Wahl der flottenabhängigen Boden- und damit auch Ankunftszeit. Ab dem flottenabhängigen Ankunftszeitpunkt von Leg l finden sich alle zulässigen Folgelegs für l wieder. Allerdings können so manche der später startenden Legs unzulässige Folgelegs sein.

Um festzustellen, ob eine gegebene Flottenzuweisung auch die verbindungsabhängigen Mindestbodenzeiten beachtet, reicht es aus, die Inseln der einzelnen Wartefunktionen unabhängig voneinander zu betrachten. Nach den Bemerkungen aus Abschnitt 4.3.1.2 müssen die Legs jeder Insel untereinander verknüpfbar sein, ohne dass dabei ein Flugzeug über die Insel hinaus auf dem Flughafen warten muss. Ohne verbindungsabhängige Mindestbodenzeiten ist dies implizit garantiert - eine FIFO-Verknüpfung führt immer zum Ziel. Mit verbindungsabhängigen Mindestbodenzeiten muss dies explizit überprüft werden.

Seien dazu für eine Insel $I = [I_S, I_E]$ von Flotte f die ankommenden Legs mit $i_1, \ldots, i_n$ bezeichnet und die abfliegenden Legs mit $o_1, \ldots, o_n$. Man beachte, dass die Anzahl der ankommenden Legs einer Insel immer gleich der Anzahl der abfliegenden Legs ist. Wir fassen

[7] Die Verknüpfung ist nur insofern unzulässig, als sie mehr Flugzeuge erfordern würde als mittels der Wartefunktionen bestimmt. Im zyklischen Fall kann ein Flugzeug immer genügend viele Planungsperioden lang auf einem Flughafen warten, um ein beliebiges von dort startendes Leg zu erreichen.

die Legs der Insel als die Knoten eines bipartiten Graphen $G = (V_i \cup V_o, E)$ auf:

$$V_i = \{i_1, \ldots, i_n\}$$
$$V_o = \{o_1, \ldots, o_n\}$$

Ein ankommendes Leg i_k ist dabei genau dann mit einem abfliegenden Leg o_l verbunden, wenn diese Legs innerhalb der Insel nacheinander von einem Flugzeug der Flotte f bedient werden können:

$$(i_k, o_l) \in E \Leftrightarrow t^d_{o_l,f} \in [t^a_{i_k,f}, I_E] \wedge g_{i_k,f} + (t^d_{o_l,f} \ominus_{\mathcal{T}} t^a_{i_k,f}) \geq g_{i_k,o_l,f}$$

Leg o_l muss zum einen nach dem Ankunftszeitpunkt von Leg i_k in der Insel starten, und zum anderen muss die dabei auftretende Bodenzeit mindestens so groß wie die verbindungsabhängige Mindestbodenzeit zwischen den Legs i_k und o_l sein.

Offensichtlich gilt nun:

Satz 4.12. *Die Legs einer Insel lassen sich genau dann innerhalb der Insel miteinander verknüpfen, wenn der oben konstruierte bipartite Graph ein perfektes Matching besitzt.*

Die Erweiterung besteht folglich darin, dass sich zu jeder Insel einer aktuellen Lösung ein perfektes Matching der ein- und ausgehenden Legs gehalten wird. Die initiale Berechnung kann mittels eines Max-Flow-Algorithmus in Zeit $O(n^{2.5})$ für Inseln mit n eingehenden Legs durchgeführt werden [Even and Tarjan, 1975, Hopcroft and Karp, 1973]. Geschwindigkeitsfördernd kommt dabei zum Tragen, dass die Berechnung nicht auf allen Ereignissen eines Flughafens gleichzeitig zu erfolgen braucht, sondern dass man inselweise vorgehen kann.

Das Update nach einem Nachbarschaftsübergang arbeitet sogar noch schneller. Hierbei werden *pro betroffener Insel* normalerweise nur zwei Legs gelöscht bzw. eingefügt, so dass ein Großteil des existierenden Matchings wiederverwendet werden kann. Ein oder zwei Flusserweiterungsschritte mit Aufwand $O(n^2)$ reichen aus, um das perfekte Matching wiederherzustellen bzw. um festzustellen, dass kein perfektes Matching existiert und der Nachbarschaftsübergang unzulässig ist. Darüber hinaus sind von einem Übergang selten mehr als 10 Inseln von Änderungen betroffen, so dass insgesamt der Test auf Zulässigkeit nach einem Nachbarschaftübergang sehr effizient durchgeführt werden kann.

Mit dieser Erweiterung ist damit sichergestellt, dass, vorausgesetzt man startet mit einer zulässigen Zuweisung, die Heuristiken auch Probleminstanzen mit verbindungsabhängigen Mindestbodenzeiten optimieren können und dabei garantieren können, nur zulässige Lösungen zu produzieren.

4.6 Homogenität

Eine häufig gewünschte Eigenschaft von Flottenzuweisungen für mehrtägige Planungsperioden ist, dass für Gruppen von Legs *nach Möglichkeit* gleiche bzw. ähnliche Flugzeugtypen eingesetzt werden sollen.

In einer Wochenplanung startet beispielsweise täglich um 10:00 ein Leg von Frankfurt nach New York. Es wäre nun schön, wenn in einer Flottenzuweisung möglichst viele dieser Legs von einem Flugzeugtyp bedient würden, da dies den an die Kunden herauszugebenden Flugplan übersichtlicher und attraktiver macht, die Planung auf dem Boden vereinfacht, unter Umständen in New York nur Wartungspersonal für einen Flugzeugtyp erforderlich macht, usw.

Es handelt sich dabei nicht um eine harte Einschränkung sondern eine Bedingung, die nur insoweit berücksichtigt werden sollte, dass sie keine zu hohen Gewinnausfälle verursacht. Dazu definieren wir

Definition 4.13 (Homogenität). *Sei ein Flottenzuweisungsproblem mit Flugplan $\mathcal{L}$ und Flotten $\mathcal{F}$, eine Zuweisung $z : \mathcal{L} \to \mathcal{F}$ für den Flugplan und eine Menge von Leggruppen $G_i \subseteq \mathcal{L}$ ($i \in \{1, \ldots, n\}$) gegeben.*

Die Homogenität einer Leggruppe G_i ist definiert durch

$$H(G_i) = |G_i| - \max_{f \in \mathcal{F}} |\{l \in G_i \mid z(l) = f\}|$$

und die Homogenität der Zuweisung durch

$$H(z) = \sum_{i=1}^{n} H(G_i)$$

$H(G_i)$ entspricht damit der minimalen Anzahl an Legs, deren Zuweisung man ändern muss, damit die Legs der Leggruppe G_i von nur einem Flugzeugtyp bedient werden. Legs, deren Zuweisung dafür geändert werden muss, werden inhomogene Legs genannt. Ein Wert von Null für $H(G_i)$ sagt somit aus, dass den Legs der Leggruppe nur ein Flottentyp zugewiesen worden ist. $H(z)$ ist ein Maß für die Homogenität einer Zuweisung - je kleiner $H(z)$, desto homogener die Leggruppen.

Damit unsere Lösungsverfahren nun möglichst homogene Zuweisungen produzieren, erweitern wir die Zielfunktion um Strafkosten für Inhomogenität

$$\text{Maximiere } P(z) - c_H H(z),$$

wobei $c_H > 0$ die Strafkosten pro inhomogenem Leg bezeichnen. Ein großer Wert für c_H wird zu sehr homogenen optimalen Zuweisungen führen, ein kleiner Wert zu Zuweisungen mit großem Gewinn, wobei homogene, fast gewinnmaximale Zuweisungen bevorzugt werden.

4.6.1 Für Lokale Suche Heuristiken

Da die Inhomogenität einer Zuweisung sehr einfach und schnell *algorithmisch* berechnet werden kann, lassen sich bei der Wahl eines Nachbarn neben den Gewinnänderungen auch die Änderungen der Inhomogenitätskosten einfach berücksichtigen.

4.6.2 Für lineare Programme

Im Gegensatz zu den Lokale Suche Heuristiken bereitet das Berücksichtigen der Homogenität von Zuweisungen in unseren linearen Programmen für das Flottenzuweisungsproblem einige Schwierigkeiten. Die Homogenität ist wegen der Maximumsbildung eine konkave Eigenschaft, die sich nur aufwendig linearisieren lässt.

Wir stellen daher drei Alternativen zur Berücksichtigung von homogenen Lösungen in unseren linearen Programmen vor - jede mit unterschiedlichen Vor- und Nachteilen. Zwei Ansätze sind exakt, ein Ansatz arbeitet nur heuristisch und kann damit die Optimalität seiner Lösung nicht garantieren.

4.6.2.1 Modell mit wenigen zusätzlichen Variablen

Bei der Beschreibung der exakten Ansätze betrachten wir nur die Modellierung, die für die Bestimmung der Inhomogenitätskosten einer Leggruppe nötig ist. Da diese Berechnung unabhängig von anderen Leggruppen ist, muss anschließend nur für jede Leggruppe ein entsprechendes Modell dem Gesamtmodell für die Flottenzuweisung hinzugefügt werden.

Für eine Leggruppe G_i ergibt sich folgende Problemstellung:

Modell 4.14.

$$\textit{Maximiere} \sum_{l \in G_i} \sum_{f \in \mathcal{F}_l} p_{l,f} y_{l,f} - c_H h \tag{4.32}$$

unter den Nebenbedingungen

$$\sum_{f \in \mathcal{F}_l} y_{l,f} = 1 \qquad \forall l \in G_i \tag{4.33}$$

$$h = |G_i| - \max_{f \in \mathcal{F}} \left(\sum_{l \in G_i : f \in \mathcal{F}_l} y_{l,f} \right) \tag{4.34}$$

$$h \geq 0 \tag{4.35}$$

$$y_{l,f} \in \{0,1\} \qquad \forall l \in G_i, f \in \mathcal{F}_l \tag{4.36}$$

Die Zielfunktion (4.32) maximiert den Gewinn der Zuweisung abzüglich der Inhomogenitätskosten. Die Gleichungen (4.33) stellen sicher, dass jedem Leg genau eine Flotte zugewiesen wird und entsprechen genau den Überdeckungsgleichungen im Time Space Network Modell. Für das Connection Network Modell muss man $y_{l,f}$ einfach überall durch $\sum_{m \in A_{l,f}} x_{l,m,f}$ ersetzen. Gleichung (4.34) berechnet die Inhomogenität der Zuweisung. Dabei handelt es sich im Moment noch nicht um eine lineare Bedingung.

Wir formulieren Gleichung (4.34) daher wie folgt um:

$$\bigvee_{f \in \mathcal{F}} \quad h + \sum_{l \in G_i : f \in \mathcal{F}_l} y_{l,f} \geq |G_i| \tag{4.37}$$

Nur eine der Ungleichungen (4.37) muss erfüllt werden. In einer optimalen Lösung sei dies die Ungleichung von Flotte f. Wegen $c_H > 0$ wird nun h in der optimalen Lösung so klein wie möglich gewählt, also so, dass die Ungleichung (4.37) für Flotte f mit Gleichheit erfüllt wird. Aus demselben Grund wird in einer optimalen Lösung f so gewählt, dass $\sum_{l \in G_i : f \in \mathcal{F}_l} y_{l,f}$ maximal ist, da dadurch h möglichst klein werden kann. Somit ist (4.37) für $c_H > 0$ äquivalent zu (4.34).

Disjunktionen der Form (4.37) sind in linearen Programmen nicht erlaubt. Sie lassen sich aber linearisieren, indem man sie durch

$$\sum_{f \in \mathcal{F}} b_f = 1 \tag{4.38}$$

$$h + \sum_{l \in G_i : f \in \mathcal{F}_l} y_{l,f} \geq b_f |G_i| \qquad \forall f \in \mathcal{F} \tag{4.39}$$

$$b_f \in \{0,1\} \qquad \forall f \in \mathcal{F} \tag{4.40}$$

ersetzt.

Daraus ergibt sich also das folgende ganzzahlige lineare Programm zur Berechnung von Zuweisungen an eine Leggruppe, die auch Inhomogenitätskosten berücksichtigt:

Modell 4.15 (HLOW).

$$\textit{Maximiere} \sum_{l \in G_i} \sum_{f \in \mathcal{F}_l} p_{l,f} y_{l,f} - c_H h \tag{4.41}$$

unter den Nebenbedingungen

$$\sum_{f \in \mathcal{F}} b_f = 1 \tag{4.42}$$

$$\sum_{f \in \mathcal{F}_l} y_{l,f} = 1 \qquad \forall l \in G_i \tag{4.43}$$

$$h + \sum_{l \in G_i : f \in \mathcal{F}_l} y_{l,f} \geq b_f |G_i| \qquad \forall f \in \mathcal{F} \tag{4.44}$$

$$h \geq 0 \tag{4.45}$$

$$b_f \in \{0,1\} \qquad \forall f \in \mathcal{F} \tag{4.46}$$

$$y_{l,f} \in \{0,1\} \qquad \forall l \in G_i, f \in \mathcal{F}_l \tag{4.47}$$

Für jede Leggruppe müssen damit zusätzlich $1 + |\mathcal{F}|$ Variablen ((4.45), (4.46)) und $1 + |\mathcal{F}|$ Bedingungen ((4.42), (4.44)) dem linearen Programm für die Flottenzuweisung hinzugefügt werden.

Das große Manko von Modell 4.15 ist, dass *in der LP-Relaxierung* die Inhomogenität h immer Null ist, also keinerlei Inhomogenitätskosten auftreten. Dadurch erhält man sehr schlechte Schranken im Branch&Bound-Löser und keine verlässliche Rückmeldung an die Baumsuche, welche b_f-Variablen wie fixiert werden sollen.

Satz 4.16. *Für die LP-Relaxierung von Modell 4.15 gilt:*

Für jede zulässige Belegung der $y_{l,f}$-Variablen existieren b_f-Zuweisungen, so dass $h = 0$ gewählt werden kann.

Beweis. Setzt man

$$b_f = \frac{\sum_{l \in G_i : f \in \mathcal{F}_l} y_{l,f}}{|G_i|} \qquad \forall f \in \mathcal{F}$$
$$h = 0$$

kann man leicht überprüfen, dass alle Bedingungen der LP-Relaxierung von Modell 4.15 erfüllt sind. □

4.6.2.2 Höherdimensionales Modell

Wir stellen nun eine alternative Formulierung zur Berechnung der Inhomogenitätskosten vor, deren LP-Relaxierung nur ganzzahlige Ecken besitzt und somit scharfe Schranken produziert. Wir leiten dabei dieses Modell von Ideen von Balas zur Disjunktiven Programmierung ab:

Satz 4.17 ([Balas, 1998]). *Gegeben $P_i = \{x \in \mathbb{R}^n : A^i x \geq b^i\} \neq \emptyset$, $i \in Q$. Die Menge $P_Q = \text{conv} \bigcup_{i \in Q} P_i$ ist die Menge der $x \in \mathbb{R}^n$, für die Vektoren $(y^i, y_0^i) \in \mathbb{R}^{n+1}$, $i \in Q$, existieren, so dass $(x, (y^i, y_0^i)_{i \in Q})$ zu $\mathfrak{P} =$*

$$\sum_{i \in Q} y^i = x \tag{4.48}$$
$$\sum_{i \in Q} y_0^i = 1 \tag{4.49}$$
$$A^i y^i \geq b^i y_0^i \qquad \forall i \in Q \tag{4.50}$$
$$y_0^i \geq 0 \qquad \forall i \in Q \tag{4.51}$$

gehört. Es gilt ferner:

- *Falls x^* eine Ecke von P_Q ist, dann ist $(x^*, (\bar{y}^i, \bar{y}_0^i)_{i \in Q})$ eine Ecke von $\mathfrak{P}$ mit $(\bar{y}^k, \bar{y}_0^k) = (x^*, 1)$ für ein $k \in Q$ und $(\bar{y}^i, \bar{y}_0^i) = (0, 0)$ für $i \in Q \setminus \{k\}$.*
- *Falls $(\bar{x}, (\bar{y}^i, \bar{y}_0^i)_{i \in Q})$ eine Ecke von $\mathfrak{P}$ ist, dann ist $\bar{x} = \bar{y}^k$ und $\bar{y}_0^k = 1$ für ein $k \in Q$, $(\bar{y}^i, \bar{y}_0^i) = (0, 0)$ für $i \in Q \setminus \{k\}$ und $\bar{x}$ eine Ecke von P_k.*

Der Satz 4.17 beschreibt, wie man die *Vereinigung* von Polytopen durch eine Transformation in einen höherdimensionalen Raum bilden kann, ohne dabei die Informationen über ihre Ecken zu verlieren. Desweiteren werden dabei *keine* ganzzahligen Variablen benötigt.

Der Lösungsraum von Modell 4.14 lässt sich wegen Bedingung (4.37) als Vereinigung der Polytope $P_t =$

$$\sum_{f \in \mathcal{F}_l} y_{l,f} = 1 \qquad \forall l \in G_i \tag{4.52}$$

$$h + \sum_{l \in G_i : t \in \mathcal{F}_l} y_{l,t} \geq |G_i| \tag{4.53}$$

$$h \geq 0 \tag{4.54}$$

$$y_{l,f} \in \{0,1\} \qquad \forall l \in G_i, f \in \mathcal{F}_l \tag{4.55}$$

mit $t \in \mathcal{F}$ beschreiben. Die P_t definierende Matrix ist vollständig unimodular, da sie, nachdem man Ungleichung (4.53) mit -1 multipliziert hat, in jeder Spalte maximal zwei {-1,1}-Einträge aufweist und jede Spalte mit zwei nicht-Null Einträgen die Spaltensumme Null besitzt. Man kann also Bedingung (4.55) durch

$$y_{l,f} \geq 0 \qquad \forall l \in G_i, f \in \mathcal{F}_l \tag{4.56}$$

ersetzen, ohne die konvexe Hülle von P_t zu verändern.

Nach Satz 4.17 lässt sich damit unser Problem wie folgt formulieren:

Modell 4.18.

$$\textit{Maximiere} \sum_{l \in G_i} \sum_{f \in \mathcal{F}_l} p_{l,f} \left(\sum_{t \in \mathcal{F}} y_{l,f}^t \right) - c_H \left(\sum_{t \in \mathcal{F}} h^t \right) \tag{4.57}$$

unter den Nebenbedingungen

$$\sum_{t \in \mathcal{F}} b_t = 1 \tag{4.58}$$

$$\sum_{f \in \mathcal{F}_l} y_{l,f}^t = b_t \qquad \forall l \in G_i, t \in \mathcal{F} \tag{4.59}$$

$$h^t + \sum_{l \in G_i : t \in \mathcal{F}_l} y_{l,t}^t \geq b_t |G_i| \qquad \forall t \in \mathcal{F} \tag{4.60}$$

$$b_t \geq 0 \qquad \forall t \in \mathcal{F} \tag{4.61}$$

$$h^t \geq 0 \qquad \forall t \in \mathcal{F} \tag{4.62}$$

$$y_{l,f}^t \geq 0 \qquad \forall l \in G_i, f \in \mathcal{F}_l, t \in \mathcal{F} \tag{4.63}$$

Man beachte, dass bei dieser Konstruktion *keine* ganzzahligen Variablen mehr benötigt werden, die LP-Relaxierung also schärfstmöglich ist. Allerdings erkauft man sich diesen Vorteil, durch eine um den Faktor $|\mathcal{F}|$ höhere Anzahl an Variablen und Bedingungen: Es werden $(2 + |G_i| \cdot |\mathcal{F}|) \cdot |\mathcal{F}|$ Variablen und $1 + |\mathcal{F}| + |G_i| \cdot |\mathcal{F}|$ Bedingungnen benötigt.

Im konkreten Fall lässt sich allerdings aus Modell 4.18 ein erheblich kleineres, äquivalentes Modell mit weniger Variablen ableiten:

Modell 4.19 (HBAL).

$$\textit{Maximiere} \sum_{l \in G_i} \sum_{f \in \mathcal{F}_l} p_{l,f} y_{l,f} - c_H \left(\sum_{f \in \mathcal{F}} h_f \right) \tag{4.64}$$

unter den Nebenbedingungen

$$\sum_{f \in \mathcal{F}} b_f = 1 \tag{4.65}$$

$$\sum_{f \in \mathcal{F}_l} y_{l,f} = 1 \qquad \forall l \in G_i \tag{4.66}$$

$$h_f + \sum_{l \in G_i : f \in \mathcal{F}_l} y^*_{l,t} \geq b_f |G_i| \qquad \forall f \in \mathcal{F} \tag{4.67}$$

$$y^*_{l,f} \leq b_f \qquad \forall l \in G_i, f \in \mathcal{F}_l \tag{4.68}$$

$$y^*_{l,f} \leq y_{l,f} \qquad \forall l \in G_i, f \in \mathcal{F}_l \tag{4.69}$$

$$b_f \geq 0 \qquad \forall f \in \mathcal{F} \tag{4.70}$$

$$h_f \geq 0 \qquad \forall f \in \mathcal{F} \tag{4.71}$$

$$y_{l,f}, y^*_{l,f} \geq 0 \qquad \forall l \in G_i, f \in \mathcal{F}_l \tag{4.72}$$

Modell 4.19 erzeugt $(2+|G_i|) \cdot |\mathcal{F}|$ zusätzliche Variablen und $1+|\mathcal{F}|+2|G_i| \cdot |\mathcal{F}|$ zusätzliche Bedingungen pro Leggruppe.

Lemma 4.20. *Für die Modelle 4.18 und 4.19 gilt:*

- *Wenn $(y^t_{l,f}, h^t, b_t)$ zum Lösungsraum von Modell 4.18 gehört, dann gehört $(y_{l,f}, y^*_{l,f}, h^t, b_t)$ zum Lösungsraum von Modell 4.19, wobei $y_{l,f} = \sum_{t \in \mathcal{F}} y^t_{l,f}$ und $y^*_{l,f} = y^f_{l,f}$ für alle $l \in \mathcal{L}$, $f \in \mathcal{F}_l$ ist.*
- *Wenn $(y_{l,f}, y^*_{l,f}, h^t, b_t)$ eine optimale Lösung von Modell 4.19 ist, dann existieren $y^t_{l,f}$, so dass $(y^t_{l,f}, h^t, b_t)$ zum Lösungsraum von Modell 4.18 gehört, und $\sum_{t \in \mathcal{F}} y^t_{l,f} = y_{l,f}$ und $y^f_{l,f} = y^*_{l,f}$ für alle $l \in \mathcal{L}$, $f \in \mathcal{F}_l$ ist.*

Beweis. Den ersten Punkt rechnet man leicht nach.

Für den zweiten Punkt gilt, dass in einer optimalen Lösung $(y_{l,f}, y^*_{l,f}, h^t, b_t)$ für Modell 4.19 wegen $c_H > 0$ alle Ungleichungen (4.67) mit Gleichheit erfüllt sind und für alle $l \in \mathcal{L}$, $f \in \mathcal{F}_l$ (mindestens) eine der Ungleichungen (4.68) bzw. (4.69) mit Gleichheit erfüllt ist, also $y^*_{l,f} = \min\{y_{l,f}, b_f\}$ gilt.

Der zweite Punkt ist also bewiesen, wenn wir für jedes Leg l die $y^t_{l,f}$-Werte so festlegen können, dass

1. $\sum_{f \in \mathcal{F}_l} y^t_{l,f} = b_t$ für alle $t \in \mathcal{F}$ (Bedingung (4.59)),
2. $\sum_{t \in \mathcal{F}} y^t_{l,f} = y_{l,f}$ für alle $f \in \mathcal{F}_l$ und

3. $y^f_{l,f} = \min\{y_{l,f}, b_f\}$ für alle $f \in \mathcal{F}_l$

gilt.

Wir initialisieren dazu eine $|\mathcal{F}| \times |\mathcal{F}|$-Matrix M mit Null. Eintrag $M(f,t)$ repräsentiert den Wert $y^t_{l,f}$. Sei ferner

$$M(f,*) = y_{l,f} - \sum_{t\in\mathcal{F}} M(f,t)$$
$$M(*,t) = b_t - \sum_{f\in\mathcal{F}} M(f,t)$$
$$F = \sum_{f\in\mathcal{F}} M(f,*)$$
$$T = \sum_{t\in\mathcal{F}} M(*,t)$$

Zu Beginn gilt offensichtlich

$$F = \sum_{f\in\mathcal{F}} M(f,*) = \sum_{f\in\mathcal{F}} y_{l,f} = 1 = \sum_{t\in\mathcal{F}} b_t = \sum_{t\in\mathcal{F}} M(*,t) = T$$

Unter dem *Setzen eines Eintrags* $M(f,t)$ verstehen wir die Zuweisung von $\min\{M(f,*), M(*,t)\}$ an $M(f,t)$. Dadurch wird

- $M(f,*)$ oder $M(*,t)$ zu Null und
- F und T nehmen um den selben Betrag $M(f,t)$ ab, sind also insbesondere anschließend immer noch gleich.

Zuerst setzen wir nun alle Elemente $M(f,f)$ der Hauptdiagonalen von M. Dadurch wird Punkt 3. erfüllt, da dieses Setzen unabhängig voneinander geschehen kann. Solange dann noch eine $M(f,*) > 0$ existiert, suchen wir ein $M(*,t) > 0$ (ein solches muss wegen $F = T$ existieren) und setzen $M(f,t)$. Dieser Eintrag kann vorher noch nicht gesetzt worden sein, da ansonsten $M(f,*)$ oder $M(*,t)$ bereits Null gewesen wäre. Nach spätestens $2|\mathcal{F}|$ Setzungen gilt $F = T = 0$ und M erfüllt auch die Punkte 1. und 2. □

Satz 4.21. *Jede optimale Ecke vom Lösungsraum von Modell 4.19 ist ganzzahlig und entspricht einer optimalen Lösung von Modell 4.14.*

Beweis. Sei $e^* = (y_{l,f}, y^*_{l,f}, h^t, b_t)$ ein optimale Ecke von Modell 4.19. Falls e^* nicht ganzzahlig ist, ist auch die nach Lemma 4.20 zu e^* gehörende Lösung $e' = (y^t_{l,f}, h^t, b_t)$ von Modell 4.18 nicht ganzzahlig. Nach Satz 4.17 ist aber das Polytop von Modell 4.18 ganzzahlig und e' kann keine Ecke sein, ist also eine echte Linearkombination von (ganzzahligen) Ecken $p'_1, \ldots, p'_k$ des Lösungsraums von Modell 4.18. Die gemäß Lemma 4.20 zu den Ecken $p'_1, \ldots, p'_k$ gehörenden Punkte $p_1, \ldots, p_k$ von Modell 4.19 sind dann aber ebenfalls ganzzahlig, mithin von e^* verschieden und liefern eine Linearkombination für e^*. Dies steht im Widerspruch zu der Annahme, dass e^* eine Ecke ist. e^* muss demnach ganzzahlig sein.

Die Ecken von Modell 4.18 korrespondieren nach Satz 4.17 mit den zulässigen Lösungen von Modell 4.14. Damit tun dies auch die optimalen Ecken von Modell 4.19. □

4.6.2.3 Heuristisch

Die Bestimmung der Inhomogenitätskosten einer Leggruppe G wäre trivial, wenn die Flotte f^*, die die meisten Legs in der Gruppe fliegt, vorab bekannt wäre. Dann bräuchten nur die Gewinne $p_{l,f}$ wie folgt angepasst werden:

$$p_{l,f}^{new} = p_{l,f} - c_H \qquad \forall l \in G, f \neq f^*$$

Es wären keine weiteren Variablen oder Bedingungen notwendig.

Die Heuristische Berücksichtigung der Inhomogenitätskosten funktioniert nun folgendermaßen:

- Löse die LP-Relaxierung des Flottenzuweisungsmodells ohne Berücksichtigung von Inhomogenitätskosten.
- Bestimme für jede Leggruppe G die Flotte f^* mit

  $$\sum_{l \in \mathcal{L}} y_{l,f^*} = \max_{f \in \mathcal{F}} \left(\sum_{l \in \mathcal{L}} y_{l,f} \right)$$

  Modifiziere wie oben beschrieben die Gewinne $p_{l,f}$ für die Legs der Leggruppe.
- Löse anschließend das Flottenzuweisungsproblem mit der modifizierten Gewinnfunktion.

Bei diesem Ansatz wird die Modellgröße durch die Inhomogenitäten nicht verändert. Es muss nur eine zusätzliche LP-Relaxierung berechnet werden. Bei dieser Vorgehensweise ist klar, dass der Wert der zurückgelieferten Zuweisung nur eine untere Schranke auf den maximalen Gewinn darstellt - man kann ja beim Festlegen auf eine Flotte f^* für eine Leggruppe G die falsche Wahl getroffen haben, was zu unnötig hohen Inhomogenitätskosten führt.

4.7 Preprocessing

4.7.1 Zulässigkeitstests

Mit ein paar einfachen Tests lässt sich vor der eigentlichen Optimierung in vielen Fällen überprüfen, ob eine gegebene Flottenzuweisungsinstanz überhaupt zulässige Lösungen besitzen kann.

Zum einen muss dafür natürlich gelten:

- Für alle Legs $l \in \mathcal{L}$: $\mathcal{F}_l \neq \emptyset$
- Für zyklische Instanzen müssen sämtliche Flughäfen bezüglich Starts und Landungen balanciert sein.

Ansonsten ist der Hauptgrund, weshalb eine Instanz keine zulässige Lösung besitzt, dass die Flugzeuganzahl nicht ausreicht, um alle Legs zu bedienen. Glücklicherweise lässt sich leicht eine in der Praxis sehr zuverlässige untere Schranke für die Gesamtanzahl an Flugzeugen ausrechnen, die mindestens für eine gegebene Instanz benötigt werden. Vergleicht man diese Schranke mit der tatsächlichen Flugzeuganzahl über alle Flotten, kann man zu kleine Flotten sehr zuverlässig vor der eigentlichen Optimierung erkennen. Dazu wird eine neue Flotte, *virtuelle Flotte* genannt, eingeführt und geschaut, wie viele Flugzeuge diese eine Flotte für den Flugplan benötigt. Ist die virtuelle Flotte passend definiert, ergibt sich so eine untere Schranke für mehrere Flotten [Gu et al., 1994].

4.7.1.1 Flugzeuganzahl bei flottenabhängigen Mindestbodenzeiten

Hier verwenden wir das Verfahren aus Satz 3.12. Die virtuelle Flotte $\hat{f}$ wird dabei folgendermaßen definiert:

- Flotte $\hat{f}$ kann alle Legs bedienen.
- Alle Mindestbodenzeiten von Flotte $\hat{f}$ sind Null.
- Für alle Legs l wird die Blockzeit für Flotte $\hat{f}$ wie folgt definiert:

$$b_{l,\hat{f}} = \min_{f \in \mathcal{F}} (b_{l,f} + g_{l,f})$$

So ist sichergestellt, dass jeder Flugzeugumlauf einer realen Flotte auch von der virtuellen Flotte übernommen werden kann, und die benötigte Flugzeuganzahl für die virtuelle Flotte stellt eine untere Schranke auf die Gesamtflugzeuganzahl aller Flotten dar.

4.7.1.2 Flugzeuganzahl bei verbindungsabhängigen Mindestbodenzeiten

Hier verwenden wir die Verfahren aus den Sätzen 3.14 und 3.16. Die virtuelle Flotte $\hat{f}$ wird dabei folgendermaßen definiert:

- Flotte $\hat{f}$ kann alle Legs bedienen.
- Alle Blockzeiten von Flotte $\hat{f}$ sind Null.
- Für alle Legs l und möglichen Folgelegs m, $s^d_m = s^a_l$, wird die verbindungsabhängige Mindestbodenzeit für Flotte $\hat{f}$ wie folgt definiert:

$$g_{l,m,\hat{f}} = \min_{f \in \mathcal{F}} (b_{l,f} + g^*_{l,m,f})$$

So ist sichergestellt, dass jeder Flugzeugumlauf einer realen Flotte auch von der virtuellen Flotte übernommen werden kann, und die benötigte Flugzeuganzahl für die virtuelle Flotte stellt eine untere Schranke auf die Gesamtflugzeuganzahl aller Flotten dar.

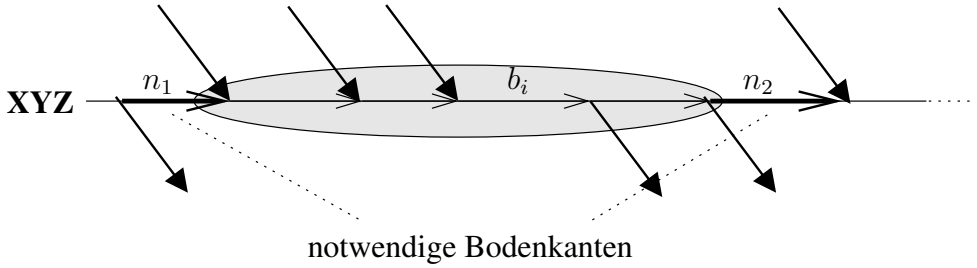


Abbildung 4.20: Flughafen mit notwendigen und nicht-notwendigen Bodenkanten. Ereignisse innerhalb der Ellipse können zusammengefasst werden.

4.7.2 Für Time Space Network Modelle

Für die Time Space Network Modelle[8] aus Abschnitt 4.2 lässt sich einfach die Anzahl an Ereignissen und damit auch Bodenkanten verringern, ohne die Menge der zulässigen Zuweisungen zu ändern.

Bodenkanten haben zwei zentrale Funktionen im Time Space Network:

- Weiterleiten von Flugzeugen von einem Ereignis eines Flughafens zum nächsten.
- Sicherstellen, dass auf einem Flughafen niemals eine negative Anzahl an Flugzeugen wartet.

Der erste Punkt lässt sich auch dann noch gewährleisten, wenn einige benachbarte Ereignisse eines Flughafen-Flotten-Paares zu einem Knoten zusammengefasst werden. Das Zusammenfassen kann dabei so geschehen, dass die Bodenzeiten von ankommenden Legs der Ereignisse so vergrößert werden und die Abflugzeiten von abfliegenden Legs so verschoben werden, dass die Ankünfte bzw. Starts auf einen Zeitpunkt fallen. Dadurch gehen dann die ursprünglichen Bodenkanten zwischen den zusammengefassten Ereignissen verloren.

Die Frage ist nun, ob es Bodenkanten gibt, für die dieses Entfernen trotzdem nicht ermöglicht, dass auf dem Flughafen eine negative Anzahl an Flugzeugen wartet. Die Antwort darauf lautet:

> Nur Bodenkanten, die von einem Flugereignis mit einem abfliegenden Leg zu einem Flugereignis mit einem ankommenden Leg führen, sind *notwendig*.

Die Begründung dafür ist folgendermaßen (siehe Abbildung 4.20). Für nicht-notwendige Bodenkanten $b_1, \ldots, b_k$, die zwischen zwei notwendigen Bodenkanten n_1 und n_2 verlaufen, gilt:

- Zu Beginn von b_1 kommt mindestens ein Leg an, da sonst n_1 keine notwendige Bodenkante wäre.

[8] Das hier Gesagte gilt auch für den Time Space Network Teil des Hybriden Modells aus Abschnitt 4.5.1.

- Am Ende von b_k startet mindestens ein Leg, da sonst n_2 keine notwendige Bodenkante wäre.
- Dazwischen dürfen bis zum Beginn von einer Bodenkante b_i zuerst nur Legs ankommen und danach nur Legs abfliegen, da ansonsten eine der Bodenkanten b_j notwendig wäre.
- Das heißt, bis zur Bodenkante b_i kann der Fluss auf den Bodenkanten $b_1, \ldots, b_i$ nur zunehmen. Wenn er für die notwendige Bodenkante n_1 nicht-negativ ist, muss er das auch für die Kanten $b_1, \ldots, b_i$ sein.
- Nach b_i kann der Fluss auf den Bodenkanten bis zu n_2 nur abnehmen. Wenn er für die notwendige Bodenkante n_2 nicht-negativ ist, muss er das auch für die vorherigen nicht-notwendigen Bodenkanten $b_i, \ldots, b_k$ sein.

Daraus folgt, dass man all die Ereignisse eines Time Space Networks zusammenfassen kann, die durch nicht-notwendige Bodenkanten miteinander verbunden sind. Dadurch wird erreicht, dass jedes Ereignis mindestens aus einem ankommenden und einem abfliegenden Leg besteht, die Anzahl der Ereignisse (und damit auch der Bodenkanten) also nicht größer als $|\mathcal{L}| \cdot |\mathcal{F}|$ sein kann.

4.7.3 Heuristisches Leg-Verschmelzen

Die Leganzahl einer Flottenzuweisungsinstanz ist der dominierende Faktor für die Laufzeit der in dieser Arbeit vorgestellten Lösungsverfahren. Dies gilt in besonderem Maße für die IP-Ansätze. Eine Möglichkeit, die Leganzahl einer Probleminstanz zu verkleinern, ist, mehrere Legs zu einem neuen Leg zu verschmelzen.

4.7.3.1 Verschmelzen von Legs

Beim Zusammenfassen von Legs geht es darum, eine Menge von Legs durch ein neues, einzelnes Leg zu ersetzen, das deren Aufgabe übernimmt. Dabei soll gewährleistet sein, dass eine zulässige Lösung, die auf einem Flugplan mit zusammengefassten Legs basiert, auch eine zulässige Lösung für den originalen Flugplan liefert und dass der Gewinn beider Lösungen identisch ist.

Um dieses zu erreichen, muss es sich bei der Menge von Legs, die zusammengefasst werden sollen, um eine Legfolge F handeln. Das neue Leg l_F hebt zur Startzeit des ersten Legs von F vom Startflughafen der Legfolge ab und landet auf deren Zielflughafen. Die Starts und Landungen auf den Zwischenstationen werden durch das neue Leg also praktisch gestrichen. Die Flugdauer des Legs l_F ergibt sich aus den Gesamtflugdauern aller zu F gehörenden Legs und den dazwischen auftretenden tatsächlichen Bodenzeiten. Entsprechend ergibt sich der Gewinn von l_F aus den Einzelgewinnen der Legs zuzüglich etwaiger verbindungsabhängiger Gewinne.

Dabei muss sichergestellt sein, dass es zumindest eine Flotte f gibt, von der ein Flugzeug die Legfolge F nacheinander bedienen kann. Flotten, die nicht dazu in der Lage sind, werden aus

der Menge der erlaubten Flotten $\mathcal{F}_{l_F}$ für Leg l_F gestrichen. Legfolgen F, für die sich $\mathcal{F}_{l_F} = \emptyset$ ergibt, werden nicht zusammengefasst, da dies ansonsten automatisch zu einer unzulässigen Flottenzuweisungsinstanz führen würde.

4.7.3.2 Konstruktion der Legfolgen

Mittels des Flugplans selbst kann man Legs finden, die vernünftigerweise von einem Flugzeug nacheinander geflogen werden sollten. „Vernünftigerweise" bedeutet in diesem Fall, dass ansonsten Flugzeuge unnötig lange auf Flughäfen warten müssten. Da Flugzeuge eine knappe und teure Ressource sind, kann dies in der Regel nicht zu einem „guten" Flugplan führen; durch das Zusammenfassen von Legs wird also in diesem Fall der Lösungsraum *in der Praxis* kaum eingeschränkt. Nichtsdestotrotz handelt es sich bei dieser Vorgehensweise um eine Heuristik.

Auf schwach frequentierten Flughäfen kommt es häufig zu der Situation, dass ein Leg ankommt, kurze Zeit später ein anderes startet und sonst für lange Zeit kein Flugereignis stattfindet. Diese Legs sind potentielle Kandidaten für eine Zusammenlegung. Würden sie nicht direkt im Anschluss von einem Flugzeug geflogen werden, hätte dies *meistens* (aber eben nicht immer) zur Folge, dass während der gesamten Planungsperiode ein Flugzeug auf dem Flughafen nutzlos wartet; dabei handelt es sich um eine Situation, die gerade auf schwach frequentierten Flughäfen nicht vorkommen sollte. Dies ist sogar teilweise eine harte Forderung, die von Fluggesellschaften an eine zulässige Flottenzuweisung gestellt wird.

Um diese Kandidaten *zuverlässig* identifizieren zu können, reicht ein Kriterium wie oben erwähnt nicht aus. Hier hilft die in Abschnitt 4.7.1.1 beschriebene Flugplananalyse mit der virtuellen Flotte.[9] Die Inseln, die sich für die virtuelle Flotte ergeben, beinhalten Legs, die nur untereinander verknüpft werden dürfen, wenn keine unnötigen Flugzeuge auf dem Flughafen der Insel warten sollen.

Für *einfache* Inseln, die nur aus einem ankommenden und einem startenden Leg bestehen, folgt so unmittelbar, dass ihre Legs nacheinander von einem Flugzeug geflogen werden müssen - die Legs sollten also verschmolzen werden. Diese Idee ist bereits von [Hane et al., 1995] eingesetzt worden, allerdings wurden dort die einfachen Inseln heuristisch bestimmt. Wir verallgemeinern diesen Ansatz jetzt für beliebige Inseln.

Ähnlich wie in Abschnitt 4.5.2 beschrieben erzeugen wir für jede Insel einen bipartiten Graphen G der ankommenden und abfliegenden Legs und verbinden Legs über eine Kante, wenn es eine (reale) Flotte gibt, die nacheinander die beiden Legs innerhalb der Insel bedienen kann. Für diesen Graphen bestimmen wir wie folgt all die Kanten, die zu *jedem* perfekten Matching gehören:

- Bestimme perfektes Matching $e_1, \ldots, e_n$ von G.
- Für jede Kante e_i:

[9] Auch im Falle von verbindungsabhängigen Mindestbodenzeiten wird für dieses heuristische Preprocessing eine virtuelle Flotte mit flottenabhängigen Mindestbodenzeiten verwendet.

- Lösche e_i aus G.
- Prüfe, ob G weiterhin ein perfektes Matching enthält. Wenn nicht, gehört e_i zu jedem perfekten Matching von G.
- Füge e_i wieder zu G hinzu.

Die Laufzeit dieses Algorithmus ist $O(n^3)$. Die initiale Berechnung eines perfekten Matching benötigt Zeit $O(n^{2.5})$. Für jede Kante e_i kann nach ihrem Löschen durch *eine* Flusserweiterung mit Laufzeit $O(n^2)$ festgestellt werden, ob weiterhin ein perfektes Matching existiert.

Gehört eine Kante zu jedem perfekten Matching, müssen die beiden betroffenen Legs nacheinander von einem Flugzeug geflogen werden, um zwischen den Inseln keine Flugzeuge auf dem zur Insel gehörenden Flughafen warten zu lassen. Die so gefundenen Kanten definieren also die Legfolgen, nach denen Legs verschmolzen werden. Insbesondere werden so auch die Verbindungen von einfachen Inseln zuverlässig erkannt.

Diese Analyse sollte wie gesagt nur für schwach frequentierte Flughäfen durchgeführt werden, da es auf stark frequentierten Flughäfen, *Hubs* genannt, durchaus erwünscht ist, dass nicht benötigte Flugzeuge dort warten. Hier kann die Wartefunktion mit ihren Inseln keine Hilfe leisten, da sie nur den Fall beschreibt, dass sich die minimal benötigte Anzahl an Flugzeugen auf dem Flughafen aufhält.

4.7.3.3 Weitere Konsequenzen für IP-Modelle

Für IP-Modelle - Connection Network, Time Space Network und Hybrides Modell - können die Informationen über die Inseln der virtuellen Flotte auf schwach frequentierten Flughäfen ferner dazu verwendet werden, nicht erwünschte Verbindungs- und Bodenkanten aus dem Modell zu entfernen. In einer Lösung, die auf den schwach frequentierten Flughäfen nicht mehr Flugzeuge benötigt wie die virtuelle Flotte, muss der Fluss auf Verbindungs- und Bodenkanten, die *zwischen* den Inseln der virtuellen Flotte verlaufen, Null sein und die entsprecheden Variablen können aus den IP-Modellen entfernt werden.

4.8 Experimentelle Ergebnisse

4.8.1 Datensätze

Für unsere Experimente standen uns 17 reale Datensätze aus den Planungsabteilungen verschiedener Fluggesellschaften zur Verfügung. Sie unterscheiden sich bezüglich ihrer Größe, Flugnetzstruktur, Erweiterungen, usw. Zu allen Datensätzen war eine zulässige Flottenzuweisung gegeben, so dass sie auch von den Lokale Suche Heuristiken optimiert werden konnten, denen ja eine zulässige initiale Lösung mitgegeben werden muss.

In Tabelle 4.1 sind die wichtigsten Merkmale der Datensätze zusammengefasst. Wir verwenden die Buchstaben A bis Q als Bezeichnung für die Datensätze. In der Spalte „Legs“ wird die Anzahl der Legs im Datensatz angegeben, in der Spalte „Flotten“ die Anzahl verschiedener

Datensatz	Legs	Flotten	Flugzeuge	Flughäfen	Flotten/Leg	azyklisch?	cdt?
A	116	2	10	18	2.0	Nein	Nein
B	288	3	27	29	3.0	Nein	Nein
C	3337	4	88	68	4.0	Nein	Nein
D	4378	11	175	75	3.2	Nein	Nein
E	4449	9	154	73	6.7	Nein	Nein
F	4449	9	154	73	6.8	Nein	Nein
G	4311	11	154	74	7.9	Nein	Nein
H	3911	10	130	63	8.9	Nein	Nein
I	4253	11	151	74	10.1	Nein	Nein
J	4565	11	159	75	10.3	Nein	Nein
K	4602	12	224	84	2.3	Nein	Ja
L	5243	23	216	76	1.7	Nein	Ja
M	6285	18	272	95	2.9	Nein	Ja
N	6287	18	261	96	2.9	Nein	Ja
O	6287	18	258	96	2.9	Nein	Ja
P	9007	9	179	74	7.9	Ja	Nein
Q	42226	13	317	148	5.4	Ja	Nein

Tabelle 4.1: Eigenschaften der 17 Probleminstanzen für die Flottenzuweisung

Flugzeugtypen, in der Spalte „Flugzeuge“ die Gesamtanzahl an verfügbaren Flugzeugen über alle Flotten und in der Spalte „Flughäfen“ die Anzahl der besuchten Flughäfen.

Bei den meisten Datensätzen können manche Legs nicht von allen Flotten bedient werden ($\mathcal{F}_l \neq \mathcal{F}$). In der Spalte „Flotten/Leg“ wird daher die durchschnittliche Anzahl an Flotten, die für ein Leg zulässig sind, angegeben. Nur bei den Datensätzen A, B und C können alle Flotten alle Legs bedienen.

Die letzten beiden Spalten klassifizieren die Datensätze genauer. Es wird angegeben, ob es sich bei dem Datensatz um eine zyklischen oder azyklische Instanz handelt (Spalte „azyklisch?“) und ob die Instanz verbindungsabhängige Mindestbodenzeiten enthält (Spalte „cdt?“). Daraus ergeben sich drei Gruppen. Die Datensätze A bis J sind zyklische Flottenzuweisungsprobleme ohne verbindungsabhängige Mindestbodenzeiten, bei den Datensätze K bis O handelt es sich um zyklische Probleminstanzen mit verbindungsabhängigen Mindestbodenzeiten und die letzten beiden Datensätze P und Q bilden die Gruppe der azyklischen Instanzen ohne verbindungsabhängige Mindestbodenzeiten.

Azyklische Flottenzuweisungsprobleme mit verbindungsabhängigen Mindestbodenzeiten standen uns nicht zur Verfügung, allerdings lässt sich jede zyklische Instanz auch als eine azyklische Instanz auffassen und als solche optimieren. In Abschnitt 4.8.8 vergleichen wir so die Performance der Verfahren für zyklische und azyklische Flottenzuweisungsprobleme.

Die größten Datensätze sind die beiden azyklischen Instanzen mit über 9000 bzw. 42000 Legs. Die Planungsperiode umfasst bei Datensatz P einen Zeitraum von zwei Wochen, bei Datensatz Q sind es sogar vier Wochen. Allen zyklischen Datensätzen ist eine Planungsdauer von einer Woche zu eigen.

Innerhalb einer jeden Gruppe sind die Datensätze nach ihrer Größe aufsteigend geordnet. Wir definieren die Größe eines Datensatzes dabei als das Produkt aus den Spalten „Legs“ und „Flotten/Leg“. Dies korrespondiert sehr genau mit der tatsächlichen Eingabegröße, da für jede *zulässige* Leg-Flotten-Kombination eine Reihe von Werten wie Blockzeit, Gewinn, usw. angegeben werden muss. Das Produkt aus Leganzahl und Flottenanzahl wäre gerade für Datensätze, bei denen Legs nicht von allen Flotten bedient werden können, nicht so aussagekräftig. So kann es passieren, dass Datensatz H als größer als Datensatz D angesehen wird, obwohl er sowohl aus weniger Legs als auch aus weniger Flotten besteht. Dafür kann in Datensatz H aber ein Leg im Durchschnitt von fast allen Flotten bedient werden.

Aus Datenschutzgründen dürfen wir bei den Ergebnissen zu den Testläufen nicht die konkret erzielten Gewinne nennen. Stattdessen geben wir die Abweichung bezüglich des optimalen Gewinns an, sofern dieser bekannt ist. Ansonsten geben wir die Abweichung bezüglich der besten bekannten oberen Schranke des Gewinns für die jeweilige Instanz an.

4.8.2 Methodik

Alle Experimente sind auf einem PC mit 3.0GHz-Pentium 4-Prozessor und 1 GB Arbeitsspeicher unter Redhat Enterprise Linux 4 durchgeführt worden. Als Lösungsverfahren für das Flottenzuweisungsproblem kamen selbstgeschriebene Programme zum Einsatz, die die Verfahren aus diesem Kapitel implementieren. Jedes der Programme steht in einer Variante für das zyklische und azyklische Flottenzuweisungsproblem zur Verfügung.

Lokale Suche Heuristiken Es sind die beiden in Abschnitt 4.3 beschriebenen Lokale Suche Verfahren auf Hill Climbing- bzw. Simulated Annealing-Basis implementiert worden. Wir kürzen hier den Hill Climbing Algorithmus mit HC und den Simulated Annealing Algorithmus mit SA ab. Alle in diesem Kapitel beschriebenen Erweiterungen sind vollständig umgesetzt worden und können bei Bedarf aktiviert werden.

In Abschnitt 4.4 werden einige Parameter, die die Nachbarschaftgenerierung steuern, beschrieben. Für unsere Testläufe haben wir bei der Generierung der Legsequenzen die Tiefensuche mit einer maximalen Tiefe von 6 und einem maximalen Grad von 4 je Knoten verwendet. Die Kandidatenauswahl in jedem Knoten erfolgt nach der BEST-Strategie (Abschnitt 4.4.1.1), das heißt, es werden unter den möglichen Kandidaten die vier gewinnträchtigsten ausgewählt. Während der Tiefensuche können mehrere zulässige Nachbarn gefunden werden und wir wählen unter diesen Nachbarn gemäß der BETTER-Strategie (Abschnitt 4.4.1.5). Somit wird die Tiefensuche beendet, sobald ein Nachbar gefunden worden ist, der von der Lokalen Suche auf jeden Fall akzeptiert wird. Damit ist die Nachbar-Generierung zielgerichtet auf das schnelle Finden von Nachbarn, die von der Lokalen Suche akzeptiert werden, ausgerichtet. Dies beschleunigt die Konvergenz der Lokale Suche Verfahren. Die hier beschriebenen Einstellungen haben sich in der Praxis bei vielen Fluggesellschaften bewährt.

Da es sich bei den beiden Lokale Suche Verfahren um randomisierte Algorithmen handelt, geben wir im Folgenden bei diesen Verfahren für die Laufzeiten und Lösungsqualitäten immer Mittelwerte aus zehn Testläufen an. Die Varianz dieser Werte über die zehn Testläufe ist im Allgemeinen so gering, dass wir auf detailliertere Angaben verzichten.

IP-basierte Verfahren Je nachdem, ob verbindungsabhängige Bodenzeiten in einer Instanz vorkommen, verwenden die IP-basierten Verfahren ein Time Space Network Modell (Abschnitt 4.2) oder ein Hybrides Modell (Abschnitt 4.5.1). Die resultierenden ganzzahligen linearen Programme werden von Standard-IP-Lösern gelöst. Dabei kann zum einen ILOG CPLEX 9.1 oder zum anderen CBC (Coin Branch and Cut solver) aus der freien COIN-OR-Library[10] zum Einsatz kommen. Mit CPLEX bezeichnen wir im Folgenden IP-Verfahren, die CPLEX als Löser einsetzen, und mit CBC die Verfahren, die CBC verwenden. Das verwendete Modell wird dabei nicht weiter spezifiziert und ergibt sich aus den Eigenschaften der Eingabeinstanz.

Beide IP-Löser basieren auf einem klassischen Branch&Bound-Ansatz, bei dem in jedem Knoten eine LP-Relaxierung des Problems gelöst wird. Liefert die LP-Relaxierung eine nichtganzzahlige Lösung, wird eine fraktionale *binäre* Variable x ausgewählt und zwei Unterprobleme mit $x = 0$ und $x = 1$ erzeugt. Wir verzweigen ausschließlich über die binäre Variablen $x_{l,m,f}$ bzw. $y_{l,f}$, da diese die eigentliche Flottenzuweisung definieren. Die Ganzzahligkeit der Bodenkantenvariablen z_{v,v^+} ergibt sich dann automatisch. Stehen mehrere fraktionale Variablen zur Auswahl, wählen wir die Variable mit dem größten Einfluss in der Zielfunktion aus, das heißt die Variable mit dem größten $p_{l,f}$-Wert.

Man kann verschiedene Baumdurchlaufstrategien während einer Branch&Bound Suche einsetzen. Wir verwenden zuerst eine Tiefensuch-Strategie, bis wir eine erste zulässige, sprich ganzzahlige Lösung gefunden haben. Dann wechseln wir zu einer Bestensuche. Wir versuchen also zuerst möglich schnell eine zulässige Lösung zu produzieren. Wegen der verwendeten Verzweigungsstrategie ist dabei die erste gefundene zulässige Lösung in der Praxis bereits von sehr hoher Qualität, weist also ein kleines integrality-gap auf.

Bereits bei mittelgroßen Flottenzuweisungsinstanzen benötigt eine vollständige Branch& Bound Suche, die beweisbar eine optimale Flottenzuweisung liefert, zu viel Zeit. Wir verwenden daher die eigentlich exakten IP-basierten Verfahren zumeist approximativ. Wir geben dabei eine relative Abweichung r vor und beenden die Branch&Bound-Suche, sobald wir eine zulässige Lösung gefunden haben, die beweisbar höchstens r vom Optimum entfernt ist. Bezeichnet P_L den Gewinn einer gefundenen Lösung L und P_{UB} die sich aus den LP-Relaxierungen der Branch&Bound-Knoten ergebene obere Schranke auf den maximalen Gewinn, so brechen wir die Suche ab, wenn gilt:

$$\frac{P_{UB} - P_L}{P_{UB}} \leq r$$

Wir verwenden typischerweise $r = 0.005$, so dass wir Lösungen produzieren, die höchstens 0.5% vom Optimum abweichen. In praktisch allen Fällen erfüllt bereits die erste gefundene ganzzahlige Lösung diese Bedingung. Berücksichtigt man dann noch, dass es sich vor allem bei den Erlöswerten in Flottenzuweisungsproblemen nur um Schätzwerte handelt, deren Fehler weit größer als 0.5% sind, reichen die so bestimmten, fast optimalen Lösungen in der Praxis vollkommen aus.

Während der Branch&Bound Suche müssen fortwährend LP-Relaxierungen gelöst werden. Das CPLEX-Verfahren verwendet dabei in der Wurzel den Barrier-Algorithmus, ein Interior-Point-Verfahren, da sich damit vor allem große LPs am schnellsten lösen lassen. Während

[10] http://www.coin-or.org

Daten-	HC		SA		CBC		CPLEX	
satz	Zeit	Qual.	Zeit	Qual.	Zeit	Qual.	Zeit	Qual.
A	0.00	opt	0.04	opt	0.01	opt	0.02	opt
B	0.02	0.52%	0.19	0.01%	0.07	opt	0.10	opt
C	3.18	1.27%	47.39	0.74%	15.34	0.00%	5.62	0.00%
D	4.61	0.87%	41.59	0.25%	61.86	0.01%	40.22	0.02%
E	6.62	1.22%	98.19	0.31%	154.40	0.00%	167.11	0.03%
F	7.13	0.58%	119.07	0.33%	100.19	0.00%	62.68	0.00%
G	8.52	0.90%	136.83	0.14%	86.08	0.00%	74.16	0.01%
H	8.06	2.40%	73.55	1.31%	60.86	0.00%	38.05	0.00%
I	17.59	1.26%	438.93	0.18%	657.17	0.02%	285.16	0.02%
J	13.89	1.08%	140.25	0.39%	154.14	0.01%	241.46	0.01%
K	5.58	0.26%	284.10	0.08%	6.36	opt	5.26	opt
L	0.37	0.52%	2.62	0.16%	3.53	0.10%	0.94	0.26%
M	9.62	3.12%	70.55	0.94%	57.39	0.00%	41.82	0.01%
N	19.63	2.82%	161.70	1.70%	234.22	0.04%	281.17	0.11%
O	6.94	2.86%	328.49	1.18%	415.59	0.08%	375.24	0.10%
P	20.66	1.27%	544.26	0.80%	1002.73	0.01%	991.82	0.01%
Q	433.15	0.45%	9090.19	0.10%	5435.33	0.00%	1938.11	0.00%

Tabelle 4.2: Laufzeit und Lösungsqualität bei maximalem Preprocessing

der Baumsuche kommt, wie bei praktisch jedem Standard-IP-Löser, der duale Simplex-Algorithmus zum Einsatz, da dieser die Basislösung des Vaterknotens als gute initiale Basis verwenden kann. In COIN-OR ist noch kein brauchbares Interior-Point-Verfahren implementiert, so dass von CBC alle Knoten mit der dualen Simplex-Methode gelöst werden.

4.8.3 Vergleich der Verfahren

In Tabelle 4.2 werden für die vier Verfahren HC, SA, CBC und CPLEX die Laufzeiten und erreichten Lösungsqualitäten auf allen 17 Probleminstanzen wiedergegeben. In der Spalte „Zeit“ ist dabei die Laufzeit des jeweiligen Verfahrens in Sekunden angegeben. Die Spalte „Qual.“ beschreibt die Qualität der von einem Verfahren produzierten Lösung. Hier wird die relative Abweichung (in Prozent) des Lösungsgewinns bezüglich der optimalen Lösung angegeben.

Für die Instanzen N bis Q ist der Wert der optimalen Lösung nicht bekannt, und hier ist die Abweichung bezüglich der besten bekannten oberen Schranke angegeben. Die Schranken stammen aus der Branch&Bound-Suche des CBC- bzw. CPLEX-Verfahrens: Die dort in den Knoten berechneten LP-Relaxierungen stellen obere Schranken auf den erzielbaren Gewinn in dem entsprechenden Unterbaum dar, und der maximale Wert einer LP-Relaxierung der noch aktiven Knoten des Suchbaums ist damit eine obere Schranke auf den Gewinn einer optimalen Lösung.

Ein Wert von beispielsweise 0.26% sagt aus, dass die vom Lösungsverfahren berechnete

Verfahren	HC	SA	CBC	CPLEX
Qualität	1.259%	0.507%	0.016%	0.034%

Tabelle 4.3: Durchschnittliche Lösungsqualität bei maximalem Preprocessing

Zuweisung mit Gewinn P_L sich zum optimalen Gewinn P^* wie folgt verhält:

$$\frac{P^* - P_L}{P^*} = 0.0026$$

Das heißt, je kleiner die Prozentzahl, desto besser ist die Lösung. Für die Instanzen, für die nur eine obere Schranke P_{UB} bekannt ist, gilt $P^* \leq P_{UB}$ und damit

$$\frac{P^* - P_L}{P^*} = 1 - \frac{P_L}{P^*} \leq 1 - \frac{P_L}{P_{UB}} = \frac{P_{UB} - P_L}{P_{UB}},$$

falls $P^* > 0$ ist, was bei allen 17 Testinstanzen der Fall ist. Also stellen in diesem Fall die Qualitätswerte obere Schranken für die tatsächliche Abweichung dar.

In den „Qual."-Spalten taucht an manchen Stellen der Eintrag „opt" auf. Dies besagt, dass das Verfahren eine optimale Lösung zurückgeliefert hat. Ein Wert von „0.00%" besagt hingegen nur, dass die Lösung *fast* optimal ist. Die relative Abweichung ist echt größer Null aber kleiner als 0.005%.

Bei allen Testläufen in Tabelle 4.2 ist vor der eigentlichen Optimierung der Datensatz mit allen (heuristischen) Preprocessing-Techniken aus Abschnitt 4.7.3 aufbereitet worden. Die Angaben zur Qualität beziehen sich auf den Wert der optimalen Lösung der aufbereiteten Instanz, *nicht* auf den optimalen Gewinn der unveränderten Instanz. In Abschnitt 4.8.4 gehen wir näher auf die Auswirkungen der verschiedenen Preprocessing-Techniken ein.

Qualität Die beiden IP-basierten Verfahren CBC und CPLEX liefern die Lösungen mit der besten Qualität. Die durchschnittliche Abweichung liegt bei nur 0.016% für CBC und 0.034% für CPLEX. Es folgt SA mit einer durchschnittlichen Abweichung von 0.507% und am Ende HC mit 1.259%.

Diese Reihenfolge zeigt sich übereinstimmend bei allen Datensätzen und ist wenig überraschend. Der Hill Climbing Algorithmus ist die einfachste Lokale Suche Heuristik und läuft Gefahr, in schlechten lokalen Optima stecken zu bleiben. Simulated Annealing versucht genau dies zu umgehen, indem vor allem anfangs auch Verschlechterungen zugelassen werden.

CBC und CPLEX sind als IP-Löser potentiell exakte Verfahren, denen hier erlaubt worden ist, die Suche bei einem relativen Fehler von 0.5% zu beenden. Bei den zurückgelieferten Lösungen handelt es sich in allen Fällen um die erste gefundene ganzzahlige Lösung im Branch&Bound-Baum. Bei realen Flottenzuweisungsproblemen treten also allem Anschein nach nur sehr kleine integrality gaps auf. Nichtsdestotrotz kann der Aufwand zum Schließen dieser Lücke, das heißt zum Bestimmen des Optimums, enorm sein, wie Abschnitt 4.8.7 zeigt.

Interessant ist die Tatsache, dass CBC in allen Fällen marginal bessere (oder zumindest gleich gute) Lösungen produziert wie CPLEX. Dabei bekommen beide IP-Löser das gleiche lineare Programm als Eingabe und auch die Baumsuchen laufen grundsätzlich ähnlich ab. Allerdings sind uns die genauen Interna von CPLEX nicht bekannt, so dass es hier vielleicht doch kleine Unterschiede zwischen CBC und CPLEX gibt, die für das Verhalten verantwortlich sind.

Es fällt auf, dass die Gewinnabweichungen für die Instanzen K bis O allgemein höher ausfallen als für die restlichen Datensätze. Dies gilt sowohl für die Heuristiken als auch die IP-basierten Verfahren. Die Instanzen K bis O enthalten verbindungsabhängige Mindestbodenzeiten, und diese scheinen das Flottenzuweisungsproblem insgesamt zu verkomplizieren.

Laufzeit Die Betrachtung der Laufzeit der Verfahren auf den Testinstanzen fällt gemischter aus. Wie nicht anders zu erwarten war, ist HC in fast allen Fällen das mit Abstand schnellste Lösungsverfahren. Die Laufzeiten der übrigen drei Verfahren SA, CBC und CPLEX bewegen sich in ähnlichen Regionen, wobei sie meistens mindestens um den Faktor 10 langsamer sind als HC. Allerdings ist hier die Streuung relativ groß.

Eine Besonderheit stellt der Datensatz K dar. Hier haben die IP-basierten Verfahren keine Schwierigkeit innerhalb von Sekunden die optimale Lösung zu berechnen, wohingegen die Lokale Suche Verfahren hart arbeiten müssen und ein Vielfaches an Laufzeit benötigen. Dafür konnten zwei Gründe ausgemacht werden. Zum einen ist die Zielfunktion der Instanz so strukturiert, dass bereits die LP-Relaxierung in der Wurzel ganzzahlig ist. Darüberhinaus sind die Gewinnwerte der Leg-Flotten-Kombinationen so beschaffen, dass dabei auch die LP-Relaxierung sehr schnell gelöst werden kann. Auf der anderen Seite starten die Lokale Suche Verfahren mit einer sehr schlechten Startlösung, so dass verhältnismäßig viele Nachbarschaftsübergänge notwendig sind, um in die Nähe des Optimums zu gelangen. Ferner sind hier die Unterschiede in den verbindungsabhängigen Mindestbodenzeiten pro Leg mit teilweise über 5 Stunden sehr groß, was es für die Nachbarschaftsgenerierung schwierig macht, überhaupt zulässige Übergänge zu finden.

Von den drei Verfahren SA, CBC und CPLEX ist CPLEX in 10 von 17 Fällen das schnellste Verfahren. Fünf Vergleiche gewinnt SA und CBC ist nur auf den beiden kleinsten Instanzen A und B am schnellsten, wo die Laufzeiten allerdings schon fast unter die Messgenauigkeit fallen. Dass der CPLEX-Optimierer CBC in fast allen Fällen schlägt, liegt hauptsächlich daran, dass die LP-Relaxierung der Wurzeln in CPLEX wegen des Einsatzes eines Interior-Point-Verfahrens um Faktoren schneller ist als in CBC mit seinem dualen Simplex-Algorithmus. Im nächsten Abschnitt 4.8.4 wird dies noch genauer herausgearbeitet.

SA kann in den meisten Fällen nur knapp mit den IP-basierten Verfahren mithalten und muss sich bei dem größten Datensatz Q deutlich geschlagen geben. Vor einigen Jahren war SA den IP-basierten Verfahren in Bezug auf die Laufzeit noch deutlich überlegen, allerdings führt die fortlaufende Verbesserung der Standard-IP-Löser dazu, dass dieser Vorsprung mittlerweile aufgeholt werden konnte. Den entscheidenden Beitrag leisten aber vor allem die verwendeten problemspezifischen Preprocessing-Techniken aus Abschnitt 4.7.3. Wir gehen im nächsten Abschnitt genauer darauf ein.

Fazit Alle Verfahren schaffen es, in akzeptabler Zeit Lösungen auch für große Flottenzuweisungsprobleme zu liefern. Die erzielten Lösungsqualitäten sind dabei insbesondere bei den Verfahren SA, CBC und CPLEX gut bis hervorragend.

Das schnellste Verfahren ist dabei HC, das auch eine Instanz mit über 40000 Legs in unter 8 Minuten optimieren kann. Allerdings kann man sich bei HC nicht darauf verlassen, immer Lösungen von ausreichender Qualität zu erhalten - sie können bis zu 3% vom Optimum abweichen, was in der Realität bei großen Fluggesellschaften Millionen-Euro-Beträge im Jahr ausmacht. HC ist daher eher für eine schnelle Beurteilung der Auswirkungen von Planänderungen geeignet.

CBC und CPLEX liefern fast optimale Lösungen und benötigen dabei auch auf großen Instanzen selten mehr als 15 Minuten Zeit. Nur bei der Instanz Q mit ihren über 40000 Legs liegen die Laufzeiten im Bereich von $1\frac{1}{2}$ bzw. $\frac{1}{2}$ Stunden. Dabei ist CPLEX häufig das etwas schnellere Verfahren, produziert dafür aber Lösungen von marginal schlechterer Qualität.

Im Vergleich zu den IP-basierten Verfahren fällt SA ein wenig zurück. Bei vergleichbaren, teils aber auch höheren Laufzeiten kann es mit einer durchschnittlichen Lösungsqualität von 0.5% nicht mit CBC bzw. CPLEX mithalten. Darüberhinaus kann SA prinzipbedingt keine Garantie auf die Lösungsqualität geben und benötigt eine zulässige Startlösung. Allerdings ist SA das einzige Verfahren, mit dem sich die in Kapitel 6 beschriebene Integration von Flottenzuweisung und Ertragsmanagement durchführen lässt. Wie die Experimente dort zeigen, lässt sich dadurch die tatsächliche Lösungsqualität nocheinmal deutlich steigern, da die dort verwendete Zielfunktion die Verhältnisse in der Realität besser abbildet.

4.8.4 Preprocessing-Techniken

Hier untersuchen wir die Auswirkungen der heuristischen Preprocessing-Techniken aus Abschnitt 4.7.3. Uns interessiert zum einen, wie sich die Preprocessing-Techniken auf die Instanzgröße, vor allem die Leganzahl, auswirkt und was dies für die Laufzeiten der verschiedenen Optimierungsverfahren bedeutet.

Bei den beschriebenen Preprocessing-Techniken handelt es sich um *Heuristiken*, die zu einem Verlust an realisierbarem Gewinn führen können. Zum anderen untersuchen wir daher diesen Effekt. Die Heuristiken gehen von der Prämisse aus, das auf schwach-frequentierten Flughäfen Flugzeuge nicht unnütz warten sollen, und können unter dieser Voraussetzung Legs auf diesen Flughäfen zusammenfassen. Wir haben bei allen Experimenten die fünf größten Flughäfen (Hubs) einer jeden Instanz von diesem Preprocessing ausgeschlossen, da auf Hubs diese Prämisse nicht gilt.

Wir unterscheiden vier Arten, nach denen die Instanzen vor der eigentliche Optimierung aufbereitet werden.

kein Preprocessing Hierbei werden die Datensätze unverändert an den Optimierer weitergegeben. Die optimale Lösung hat der größten Gewinn.

einfache Inseln Hier werden Legs zusammengefasst, die zu einfachen Inseln gehören. Dieses Verfahren wurde bereits in [Hane et al., 1995] eingesetzt.

Daten-	kein Prepro.		einfache Inseln			alle Inseln			komplett
satz	Legs	FH	Legs	FH	Δ-Gewinn	Legs	FH	Δ-Gewinn	Δ-Gewinn
A	116	18	87	7	-	87	7	-	-
B	288	29	233	13	-	233	13	-	-
C	3337	68	2293	21	-	2293	21	-	-
D	4378	75	3351	21	0.00%	3228	20	0.00%	0.00%
E	4449	73	3389	20	0.26%	3376	20	0.26%	0.29%
F	4449	73	3393	20	0.00%	3370	20	0.00%	0.00%
G	4311	74	3326	22	0.00%	3326	22	0.00%	0.06%
H	3911	63	3131	23	0.19%	3131	23	0.19%	0.38%
I	4253	74	3347	21	0.18%	3347	21	0.18%	0.27%
J	4565	75	3531	24	0.00%	3531	24	0.00%	0.00%
K	4602	84	3913	28	0.01%	3862	28	0.01%	0.01%
L	5243	76	4325	38	2.55%	4133	31	2.80%	2.82%
M	6285	95	5033	25	0.32%	4482	22	1.01%	1.31%
N	6287	96	5053	26	0.61%	4486	24	0.90%	0.99%
O	6287	96	5053	26	0.38%	4486	24	0.56%	0.61%
P	9007	74	7412	45	0.02%	7380	45	0.02%	0.31%
Q	42226	148	34116	91	0.11%	33668	91	0.12%	0.27%

Tabelle 4.4: Auswirkung von Preprocessing auf die Instanzgröße und den Gewinn

alle Inseln Alle Inseln werden darauf untersucht, ob es innerhalb der jeweiligen Insel Verbindungen gibt, die auf jeden Fall verwendet werden müssen, um keine Flugzeuge unnütz auf dem Flughafen warten zu lassen. Es stellt eine Verallgemeinerung des „einfache Insel"-Preprocessing dar.

komplett Diese Technik lässt sich nur bei IP-basierten Verfahren einsetzen und verbietet zusätzlich explizit Fluss zwischen Inseln von schwach-frequentierten Flughäfen. Die Leganzahl wird dadurch nicht weiter verringert, aber es werden weniger Variablen in den IP-Modellen benötigt.

Falls die Startlösung der heuristischen Verfahren keine unnützen Flugzeuge auf schwach-frequentierten Flughäfen einsetzt, garantieren HC und SA *implizit*, dass dies auch für die von ihnen zurückgelieferten Lösungen der Fall ist, da Nachbarschaftsübergänge nicht zu einem Mehrverbrauch an Flugzeugen auf Flughäfen führen können. Insofern wird diese Preprocessing-Variante von den Heuristiken automatisch verwendet.

Gewinneinbußen In Tabelle 4.4 werden die Auswirkungen der Preprocessing-Techniken auf die Instanzgröße und den einhergehenden Gewinnrückgang wiedergegeben. In den „Leg"-Spalten ist dabei die Anzahl an Legs, die nach dem Preprocessing übrigbleiben wiedergeben, und in den „FH"-Spalten steht, wie viele Flughäfen von diesen Legs noch angeflogen werden. Für die Preprocessing-Variante „komplett" sind diese Werte nicht angegeben, da sie sich nicht von den Werten der Variante „alle Inseln" unterscheiden.

Preprocessing	kein Prepro.	einfach Inseln	alle Inseln	komplett
Δ-Gewinn	-	0.272%	0.356%	0.431%

Tabelle 4.5: Durchschnittliche Gewinneinbußen durch Preprocessing

Die „Δ-Gewinn"-Spalten geben den Rückgang des erzielbaren optimalen Gewinns durch das jeweilige Preprocessing an. Die Angaben beziehen sich auf den relativen Verlust bezüglich des Gewinns einer optimalen Lösung der nicht-aufbereiteten Instanz. Allerdings sind diese optimalen Gewinne nur für die kleinsten Instanzen bekannt - in den anderen Fällen haben wir, wie im vorherigen Abschnitt ausführlich beschrieben, oberer Schranken auf die optimalen Gewinne verwendet. Ein Eintrag von 0.26% bedeutet demnach, dass mit der entsprechenden Preprocessing-Trechnik höchstens noch ein Gewinn in Höhe von 99.74% der Originalinstanz erzielt werden kann. Der Eintrag „-" steht dafür, dass sich der Wert einer optimalen Lösung durch das entsprechende Preprocessing nicht verringert.

Durch die Preprocessing-Techniken kann die Leganzahl einer Instanz im Durchschnitt um ein Viertel verringert werden. Ein Großteil der Einsparung wird dabei bereits durch das „einfache Insel"-Preprocessing erreicht. Die Ausweitung auf alle Inseln bringt nur für Instanzen mit verbindungsabhängigen Mindestbodenzeiten eine weitere deutliche Abnahme der Leganzahl. Bei den übrigen Instanzen wird dadurch die Leganzahl kaum oder gar nicht weiter verringert. Gerade bei verbindungsabhängigen Mindestbodenzeiten sind innerhalb von Inseln der virtuellen Flotte viele Verbindungen zwischen Legs unzulässig, so dass hier häufiger erzwungende Legverbindungen gefunden werden können.

Dass das Zusammenfassen von Legs gerade auf kleinen Flughäfen sehr effektiv ist, beweisen die Zahlen zu den angeflogenen Flughäfen der Instanzen. Ihre Anzahl sinkt durch das Preprocessing im Durchschnitt um den Faktor 3 bis 4. Die Mehrzahl der Flughäfen einer typischen Flottenzuweisungsinstanz sind schwach-frequentiert und die Legs partitionieren sich dort komplett zu einfachen Inseln.

Die durch Preprocessing zu erwartenden durchschnittlichen Gewinneinbußen liegen im Bereich von 0.3%. Dabei gibt es Instanzen, die fast gänzlich immun gegenüber Gewinneinbußen durch Preprocessing sind (z.B. Datensatz F, J und K) und andere, wo der erzielbare Gewinn um über 2% einbricht (Datensatz L). Gerade die Datensätze mit verbindungsabhängigen Mindestbodenzeiten (K bis O) scheinen stärker davon betroffen zu sein.

Unterschiede zwischen der „einfache Inseln"-Variante und der „alle Inseln"-Variante machen sich praktisch nur bei Instanzen mit verbindungsabhängigen Mindestbodenzeiten bemerkbar, da nur hier signifikant mehr Legs durch die „alle Inseln"-Variante zusammengefasst werden. Der Schritt hin zur „komplett"-Variante für IP-basierte Verfahren senkt den erzielbaren Gewinn in den meisten Fällen nochmal wahrnehmbar.

Laufzeit der IP-basierten Verfahren Kommen wir nun zu den Auswirkungen der verschiedenen Preprocessing-Techniken auf die Laufzeit der Optimierungsverfahren. Für eine repräsentative Auswahl von Instanzen sind die Ergebnisse dazu in den Tabellen 4.6 bis 4.11 zusammengefasst.

Prepro-	Datensatz C					
cessing	$y_{l,f}$	z_{v,v^+}	$x_{l,m,f}$	Spalten	Zeilen	Zeit pro Übergang (ms)
kein	13348	8821	0	22169	12162	0.17
einfache I.	8777	4684	0	13461	6981	0.19
alle I.	8777	4684	0	13461	6981	0.19
komplett	8723	4139	0	12862	6976	-

Prepro-	CBC			CPLEX			HC	SA
cessing	Wurzel	Knoten	Zeit	Wurzel	Knoten	Zeit	Zeit	Zeit
kein	77.68	11	88.84	16.82	6	26.84	3.32	35.61
einfache I.	19.71	8	22.83	3.53	7	5.55	3.33	46.87
alle I.	19.71	8	22.83	3.53	7	5.55	3.18	47.39
komplett	13.54	7	15.34	2.41	7	5.62	-	-

Tabelle 4.6: Auswirkung von Preprocessing auf die Modellgröße und Laufzeit; Datensatz C

Prepro-	Datensatz E					
cessing	$y_{l,f}$	z_{v,v^+}	$x_{l,m,f}$	Spalten	Zeilen	Zeit pro Übergang (ms)
kein	29925	16378	0	46303	20836	0.19
einfache I.	20981	9920	0	30901	13318	0.16
alle I.	20822	9847	0	30669	13232	0.17
komplett	20577	8489	0	29066	13252	-

Prepro-	CBC			CPLEX			HC	SA
cessing	Wurzel	Knoten	Zeit	Wurzel	Knoten	Zeit	Zeit	Zeit
kein	481.36	21	606.31	92.28	20	221.38	8.66	102.77
einfache I.	153.40	13	195.50	27.80	19	68.85	7.11	106.79
alle I.	136.23	12	166.89	29.95	13	54.65	6.62	98.19
komplett	99.25	41	154.40	28.17	106	167.11	-	-

Tabelle 4.7: Auswirkung von Preprocessing auf die Modellgröße und Laufzeit; Datensatz E

Preprocessing	Datensatz I					
	$y_{l,f}$	z_{v,v^+}	$x_{l,m,f}$	Spalten	Zeilen	Zeit pro Übergang (ms)
kein	42984	22932	0	65916	27196	0.21
einfache I.	30734	13748	0	44482	17106	0.20
alle I.	30734	13748	0	44482	17106	0.20
komplett	29932	11327	0	41259	17032	-

Preprocessing	CBC			CPLEX			HC	SA
	Wurzel	Knoten	Zeit	Wurzel	Knoten	Zeit	Zeit	Zeit
kein	1104.37	215	3322.23	274.35	229	3637.04	16.48	349.25
einfache I.	414.48	100	1011.00	69.10	158	956.67	18.03	423.22
alle I.	414.48	100	1011.00	69.10	158	956.67	17.59	438.93
komplett	225.13	99	657.17	49.49	50	285.16	-	-

Tabelle 4.8: Auswirkung von Preprocessing auf die Modellgröße und Laufzeit; Datensatz I

Preprocessing	Datensatz J					
	$y_{l,f}$	z_{v,v^+}	$x_{l,m,f}$	Spalten	Zeilen	Zeit pro Übergang (ms)
kein	47058	24645	0	71703	29221	0.17
einfache I.	31021	13976	0	44997	17518	0.14
alle I.	31021	13976	0	44997	17518	0.13
komplett	29678	11597	0	41275	17438	-

Preprocessing	CBC			CPLEX			HC	SA
	Wurzel	Knoten	Zeit	Wurzel	Knoten	Zeit	Zeit	Zeit
kein	375.36	87	1360.41	312.51	187	3415.89	14.85	212.43
einfache I.	112.99	137	500.93	74.17	152	522.75	13.66	122.87
alle I.	112.99	137	500.93	74.17	152	522.75	13.89	116.72
komplett	53.51	62	154.14	62.48	100	241.46	-	-

Tabelle 4.9: Auswirkung von Preprocessing auf die Modellgröße und Laufzeit; Datensatz J

Preprocessing	Datensatz M					
	$y_{l,f}$	z_{v,v^+}	$x_{l,m,f}$	Spalten	Zeilen	Zeit pro Übergang (ms)
kein	18068	11273	5217	34558	17600	0.57
einfache I.	14362	7888	4414	26664	13519	0.28
alle I.	12699	6573	4254	23526	11675	0.25
komplett	12585	5729	4239	22553	11697	-

Preprocessing	CBC			CPLEX			HC	SA
	Wurzel	Knoten	Zeit	Wurzel	Knoten	Zeit	Zeit	Zeit
kein	209.05	309	1017.93	33.02	508	1836.21	18.18	156.40
einfache I.	72.19	65	200.07	16.73	109	104.24	9.77	106.34
alle I.	33.99	70	116.60	13.90	71	55.56	9.62	70.55
komplett	17.29	50	57.39	12.85	82	41.82	-	-

Tabelle 4.10: Auswirkung von Preprocessing auf die Modellgröße und Laufzeit; Datensatz M

Preprocessing	Datensatz N					
	$y_{l,f}$	z_{v,v^+}	$x_{l,m,f}$	Spalten	Zeilen	Zeit pro Übergang (ms)
kein	18168	11369	8177	37714	17768	0.29
einfache I.	14563	7937	6245	28745	13780	0.42
alle I.	12910	6537	6092	25539	11890	0.66
komplett	12828	5872	6067	24767	11899	-

Preprocessing	CBC			CPLEX			HC	SA
	Wurzel	Knoten	Zeit	Wurzel	Knoten	Zeit	Zeit	Zeit
kein	282.21	417	2429.25	41.02	383	2067.49	6.13	93.43
einfache I.	133.79	219	627.36	18.70	367	775.31	7.79	142.97
alle I.	73.02	299	442.55	14.16	206	270.13	19.63	161.70
komplett	54.77	178	234.22	12.89	220	281.17	-	-

Tabelle 4.11: Auswirkung von Preprocessing auf die Modellgröße und Laufzeit; Datensatz N

Im oberen Teil werden für jeden betrachteten Datensatz die Auswirkung des Preprocessings auf die Modellgröße der IP-basierten Verfahren und auf die durchschnittliche Generierungszeit eines Nachbarn für die Lokale Suche Verfahren aufgezeigt. Die $y_{l,f}$-, z_{v,v^+}- und $x_{l,m,f}$-Spalten geben dabei die Anzahl entsprechender Variablen in den IP-Modellen wieder. Die folgenden „Spalten"- und „Zeilen"-Spalten fassen nochmal die Größe der resultierenden Constraint-Matrix zusammen. All diese Angaben beziehen sich natürlich auf die IP-basierten Verfahren CBC und CPLEX.

Für die Lokale Suche Verfahren HC und SA wird in der Spalte „Zeit pro Übergang" die durchschnittliche Zeit in Millisekunden für die Generierung eines Nachbarn angegeben. Für das „komplett"-Preprocessing fehlt dabei diese Angabe, da es nur bei IP-basierten Verfahren zum Einsatz kommt.

Im unteren Teil werden für jedes Verfahren und jede Preprocessing-Variante die resultierenden Lösungszeiten in Sekunden in den „Zeit"-Spalten angegeben. Für die IP-basierten Verfahren werden zusätzlich noch die Zeit zum Lösen der LP-Relaxierung in der Wurzel und die Anzahl der während der Branch&Bound-Suche besuchten Knoten angegeben. Auch hier entfallen für das „komplett"-Preprocessing die Zeiten der Lokale Suche Verfahren.

Die Ergebnisse für die IP-basierten Verfahren sind eindeutig: Je mehr Legs durch das Preprocessing zusammengefasst werden, desto weniger Variablen werden in den IP-Modellen benötigt, desto kleiner werden die Constraint-Matrizen und desto schneller lassen sich die LP-Relaxierungen lösen. Da in den meisten Fällen dabei auch die Anzahl besuchter Knoten während der Branch&Bound Suche kleiner wird, sinken die Lösungszeiten durch Preprocessing gewaltig. Einsparungen um den Faktor 10 und mehr sind gerade bei großen Instanzen üblich, wenn man die Laufzeiten der nicht-aufbereiteten und der „komplett" aufbereiteten Instanzen miteinander vergleicht.

Es fällt auf, dass die Wurzelknoten vom CBC-Verfahren um einiges langsamer gelöst werden als vom CPLEX-Verfahren. Dies ist den unterschiedlichen LP-Lösern geschuldet, die CBC bzw. CPLEX im Wurzelkonten einsetzen. Wie oben bereits erwähnt, steht in CPLEX mit dem Barrier-Algorithmus ein Interior-Point-Verfahren zur Verfügung, das sehr effizient auf unseren IP-Modellen arbeitet. Die Dauer der anschließenden Branch&Bound Suche wird im wesentlichen von der Anzahl besuchter Knoten bestimmt, da hier beide IP-Löser ähnlich effiziente duale Simplex Methoden verwenden.

Normalerweise sinken mit der Modellgröße auch die Anzahl besuchter Knoten im Branch& Bound Baum, so dass sich eine Verkleinerung der Modellgröße doppelt positiv auswirkt. Bei manchen Instanzen scheint es aber so zu sein, dass insbesondere durch ein „komplett"-Preprocessing die Knotenanzahl auch ansteigen kann - diese Preprocessing-Variante birgt also die Gefahr, die IP-Modelle strukturell komplizierter zu machen. Besonders deutlich ist dieser Effekt bei Datensatz E in Tabelle 4.7 zu beobachten.

Nichtsdestotrotz gilt im Allgemeinen, dass man für kurze Laufzeiten der IP-basierten Verfahren die Datensätze soweit wie möglich aufbereiten und verkleinern sollte. Die Algorithmen werden dadurch um Faktoren schneller.

Laufzeit der Lokale Suche Verfahren Die Lokale Suche Verfahren HC und SA reagieren auf Preprocessing nicht so eindeutig wie die IP-basierten Verfahren CBC und CPLEX.

Hier lassen sich alle möglichen Effekte beobachten: die Laufzeit kann durch Preprocessing sinken (Datensatz J und M), nahezu unverändert bleiben (Datensatz C, E und I) oder sogar ansteigen (Datensatz N).

Eine mögliche Erklärung dafür, dass sich Preprocessing nicht so günstig auf die Lokale Suche Verfahren auswirkt, ist wie folgt. Bei der Generierung der Legsequenzen für die Lokale Suche Verfahren kommt es auf schwach-frequentierten Flughäfen häufig vor, das für ein ankommendes Leg nur *ein* Folgelegkandidat existiert, der zusammen mit dem ankommenden Leg eine einfache Insel bildet. In diesem Fall wird der Tiefensuchbaum nicht besonders breit und lässt sich schnell durchsuchen. Das Preprocessing verknüpft nun aber bereits vor der eigentlichen Optimierung solche Legs, so dass für aufbereitete Instanzen der Tiefensuchbaum breiter und sogar im gewissen Sinne tiefer wird. Die produzierten Legsequenzen bestehen zwar weiterhin aus maximal sechs (ggf. zusammengefassten) Legs, durch das Zusammenfassen können sie aber effektiv mehr als sechs ursprüngliche Legs repräsentieren. So wird die Größe der Nachbarschaft einer Instanz durch Preprocessing implizit vergrößert und die Effekte durch die verringerte Leganzahl wieder aufgefressen.

Fazit Für die IP-basierten Verfahren ist ein gründliches Preprocessing immer anzuraten, da die Laufzeiten so um Faktoren gesenkt werden können. Dabei gilt generell, dass je weniger Legs und damit Variablen ein IP-Modell enthält, desto schneller lässt es sich lösen. Für lokale Suche Verfahren ist Preprocessing aus Laufzeitüberlegungen nicht so entscheidend, da es sich nur in manchen Fällen laufzeitsenkend auswirkt und im Extremfall sogar zu längeren Laufzeiten führen kann.

Demgegenüber steht der unvermeidliche Verlust an erzielbarem Gewinn durch die hier behandelten Preprocessing-Techniken. Für die meisten Instanzen hält sich dieser Verlust in engen Grenzen und spielt kaum eine Rolle in der Praxis. Es ist sogar vielmehr so, dass viele Fluggesellschaften auf schwach-frequentierten Flughäfen keine überflüssigen Flugzeuge am Boden warten haben wollen und dafür lieber auf ein paar Promille an Gewinn verzichten. Die Preprocessing-Techniken arbeiten nach dieser Prämisse und ihr Einsatz erzeugt damit implizit Flottenzuweisungen mit dieser gewünschten Eigenschaft.

4.8.5 Verbindungsabhängige Mindestbodenzeiten

In diesem Abschnitt wollen wir untersuchen, wie sich verbindungsabhängige Mindestbodenzeiten bei der Optimierung von Flottenzuweisungsproblemen auswirken. Die dafür notwendigen Anpassungen an den IP-Modellen bzw. Lokale Suche Heuristiken sind recht gravierend, so dass negative Auswirkungen auf die Laufzeit und gegebenenfalls auch die Lösungsqualität zu erwarten sind.

Um diesen Vergleich auf strukturell ähnlichen Instanzen durchführen zu können, haben wir die fünf Datensätze K bis O dupliziert und daraus Instanzen K' bis O' ohne verbindungsabhängige Mindestbodenzeiten erzeugt. Dabei kam das in Abschnitt 4.5.2 vorgestellte Verfahren zum Einsatz, mit dem bereits für die Lokale Suche Verfahren flottenabhängige Mindestbodenzeiten aus verbindungsabhängigen Mindestbodenzeiten berechnet wurden. Die Menge der zulässigen Lösungen der Instanzen K' bis O' ist damit eine Obermenge der zulässigen Lösungen der

Daten-	HC		SA		CBC		CPLEX	
satz	Zeit	Qual.	Zeit	Qual.	Zeit	Qual.	Zeit	Qual.
K	5.58	0.26%	284.10	0.08%	6.36	opt	5.26	opt
K'	3.18	0.12%	236.95	0.03%	3.25	opt	2.68	opt
L	0.37	0.52%	2.62	0.16%	3.53	0.10%	0.94	0.26%
L'	0.53	0.49%	5.44	0.36%	6.15	0.37%	1.02	0.20%
M	9.62	3.12%	70.55	0.94%	57.39	0.00%	41.82	0.01%
M'	6.29	3.73%	55.89	0.57%	34.71	0.01%	23.23	0.02%
N	19.63	2.82%	161.70	1.70%	234.22	0.04%	281.17	0.11%
N'	5.14	2.49%	74.59	1.85%	118.95	0.01%	88.72	0.04%
O	6.94	2.86%	328.49	1.18%	415.59	0.08%	375.24	0.10%
O'	6.31	2.74%	55.37	1.68%	137.11	0.03%	141.88	0.07%

Tabelle 4.12: Laufzeit und Lösungsqualität für Instanzen mit (X) und ohne (X') verbindungsabhängige Mindestbodenzeiten bei maximalem Preprocessing

Instanzen K bis O. Von der Flugplanstruktur und der Gewinnfunktion sind sich ansonsten die Instanzen so ähnlich wie nur möglich.

Tabelle 4.12 fasst die Ergebnisse dieser Untersuchung zusammen. Wie schon in Tabelle 4.2 werden für die vier Verfahren HC, SA, CBC und CPLEX die Laufzeiten und erreichten Lösungsqualitäten auf den 10 untersuchten Instanzen wiedergegeben. In der Spalte „Zeit“ ist dabei die Laufzeit des jeweiligen Verfahrens in Sekunden angegeben. Die Spalte „Qual.“ beschreibt die Qualität der von einem Verfahren produzierten Lösung. Hier wird die relative Abweichung (in Prozent) des Lösungsgewinns bezüglich der optimalen Lösung angegeben.

Die Lösungsqualitäten scheinen weitgehend unabhängig davon zu sein, ob eine Instanz mit oder ohne verbindungsabhängigen Mindestbodenzeiten gelöst werden muss. Sie bewegen sich im direkten Vergleich zweier Instanzen X und X' auf ähnlichem Niveau für jedes der betrachteten Verfahren. Mal wird für die eine Instanz eine etwas bessere Lösung produziert, mal für die andere. Es lässt sich höchstens eine leichte Tendenz ausmachen, dass die IP-basierten Verfahren für die Instanzen ohne verbindungsabhängige Mindestbodenzeiten etwas näher an die optimale Lösung gelangen können.

Ansonsten ändert sich gegenüber dem generellen Vergleich der Verfahren in Abschnitt 4.8.3 nichts: die IP-basierten Verfahren erzeugen fast optimale Lösungen, gefolgt von SA und mit Abstand dahinter HC.

Zumindest für die größeren Instanzen M bis O sinken die Laufzeiten ohne verbindungsabhängige Mindestbodenzeiten bei allen Verfahren deutlich, typischerweise um den Faktor 2 bis 3. Bei den Lokale Suche Verfahren spart man sich das vergleichsweise aufwendige Überprüfen der Gültigkeit eines Nachbarn mittels perfekter Matchings, die ansonsten ständig aktuell gehalten werden müssen. Bei den IP-basierten Verfahren kommt hauptsächlich die geringere Modellgröße zum tragen, da keine Verbindungsvariablen $x_{l,m,f}$ mehr benötigt werden.

In Tabelle 4.13 ist dies beispielhaft für die beiden Datensätze O und O' wiedergegeben. Der Aufbau der Tabelle entspricht dem der Tabellen 4.6 bis 4.11. Es werden oben die re-

Preprocessing	Datensatz O und O'					
	$y_{l,f}$	z_{v,v^+}	$x_{l,m,f}$	Spalten	Zeilen	Zeit pro Übergang (ms)
kein	18168	11369	8177	37714	17768	0.29
kein	18074	11260	0	29334	17565	0.21
einfache I.	14563	7937	6245	28745	13780	0.40
einfache I.	13791	7826	0	21617	12897	0.19
alle I.	12910	6537	6092	25539	11890	0.59
alle I.	12061	6436	0	18497	10940	0.20
komplett	12828	5872	6067	24767	11899	-
komplett	11979	5771	0	17750	10949	-

Preprocessing	CBC			CPLEX			HC	SA
	Wurzel	Knoten	Zeit	Wurzel	Knoten	Zeit	Zeit	Zeit
kein	280.30	414	1984.69	38.14	478	2622.96	6.77	91.08
kein	229.74	176	1062.85	43.22	255	1050.15	4.89	43.60
einfache I.	134.98	225	579.93	18.81	190	415.89	5.14	108.59
einfache I.	81.09	70	234.81	18.70	106	186.51	4.80	55.65
alle I.	75.86	323	535.24	18.66	376	619.04	6.94	328.49
alle I.	52.04	80	177.07	13.99	103	112.65	6.31	55.37
komplett	61.46	302	415.59	13.09	321	375.24	-	-
komplett	36.28	74	137.11	11.18	141	141.88	-	-

Tabelle 4.13: Auswirkung von verbindungsabhängigen Mindestbodenzeiten auf die Modellgröße und Laufzeit; Datensatz O und O'

sultierenden Modellgrößen und durchschnittlichen Zeiten für die Nachbargenerierung für die verschiedenen Preprocessing-Varianten angegeben. Die obere Zeile einer jeden Variante steht dabei für Datensatz O und die untere Zeile für Datensatz O'. Im unteren Teil werden entsprechend die Laufzeiten unserer vier Verfahren gegenübergestellt.

Bei den IP-Modellen fallen ohne verbindungsabhängige Mindestbodenzeiten vor allem die Verbindungsvariablen $x_{l,m,f}$ weg, was die Variablenanzahl um knapp ein Drittel reduziert. Entsprechen schneller können die LP-Relaxierungen gelöst werden, wobei im Wurzelknoten CPLEX gegenüber CBC wieder im Vorteil ist. Auffällig ist, das neben der schnelleren Lösbarkeit der LP-Relaxierungen auch deutlich weniger Knoten im Branch&Bound Baum besucht werden. Zusammengenommen ergibt sich durch die Bank ein Laufzeitvorteil um den Faktor 3 bei den IP-basierten Verfahren, wenn man auf verbindungsabhängige Mindestbodenzeiten verzichtet.

Auch die Lokale Suche Verfahren können Instanzen ohne verbindungsabhängige Mindestbodenzeiten schneller lösen. Insbesondere SA kann bei der „alle Inseln"-Preprocessing-Variante um den Faktor 6 zulegen. Bei der Instanz O handelt es sich um eine Instanz, bei der das Preprocessing negativ auf die Laufzeit der Lokale Suche Verfahren wirkt. Die durchschnittliche Zeit zum Generieren eines Nachbarschaftsübergangs verdoppelt sich mit zunehmendem Preprocessing von 0.29ms auf 0.59ms. Dieser Effekt tritt ohne verbindungsabhängige Min-

Daten-	Leg-	IP mit Hom.		IP mit HBAL		IP mit HLOW	
satz	gruppen	Spalten	Zeilen	Spalten	Zeilen	Spalten	Zeilen
A	36	269	192	645	765	377	301
B	62	1142	686	2366	2639	1390	935
C	563	12862	6976	30418	35896	15677	9792
D	696	14401	8474	77442	112285	22753	16827
E	688	29066	13252	81221	99675	35946	20133
F	688	29949	13384	75912	93615	36829	20265

Tabelle 4.14: IP-Modellgrößen für unterschiedliche Homogenitätsmodelle

destbodenzeiten nicht mehr auf. Dementsprechend sind die Laufzeiten der Instanz O' nicht besonders vom Preprocessing abhängig.

4.8.6 Homogenität

Wir untersuchen hier, was für Auswirkungen das Erstellen von homogenen Flottenzuweisungen auf die Lösungsverfahren hat. Wir haben in Abschnitt 4.6 beschrieben, wie die Lokale Suche Verfahren und die IP-basierten Verfahren heuristisch mit Homogenitätskosten umgehen können. Starten wollen wir aber mit der Untersuchung der beiden exakten Möglichkeiten für die IP-basierten Verfahren.

Exakte Modellierung Bei den beiden exakten Ansätzen zur Berücksichtigung von Homogenitätskosten werden die IP-Modelle für die Flottenzuweisung um Modelle der Form HBAL 4.19 oder HLOW 4.15 erweitert - für jede zu berücksichtigende Leggruppe muss ein Modell hinzugefügt werden. Die hinzugefügten Modelle dienen dabei ausschließlich der Berechnung der Inhomogenität einer Flottenzuweisung, um diese mit Strafkosten versehen in der Zielfunktion berücksichtigen zu können.

Tabelle 4.14 zeigt für die Datensätze A bis F, wie sich die IP-Modelle durch das Hinzufügen der Homogenitätskomponenten vergrößern. Für jeden Datensatz wird in der Spalte „Leggruppen“ die Anzahl zu berücksichtigender Leggruppen angegeben - entsprechend viele Teilmodelle müssen dem Flottenzuweisungsmodell hinzugefügt werden. Die Spalten mit der Überschrift „IP mit Hom.“ beschreiben die IP-Modellgröße für die heuristische Variante zur Berücksichtigung von Homogenitätskosten, bei der nur die Zielfunktion des Ursprungsmodells verändert wird und keine neuen Komponenten hinzugefügt werden. Die Spalten mit den Überschriften „IP mit HBAL“ und „IP mit HLOW“ geben die Modellgrößen für die beiden exakten Modellierungen an.

Beide Varianten HBAL und HLOW vergrößern die IP-Modelle deutlich, wobei HBAL nochmal deutlich mehr Variablen und vor allem Gleichungen dem Ursprungsmodell hinzufügt. Nach den Ausführungen in den Abschnitten 4.6.2.1 und 4.6.2.2 ist dies nicht überraschend.

In Tabelle 4.15 sind die Laufzeiten und erzielten Lösungsqualitäten des CPLEX-Optimierers mit den unterschiedlichen Varianten zur Homogenitätsbestimmung wiedergegeben. Zum Ver-

Daten-	CPLEX ohne Hom.		CPLEX mit Hom.		CPLEX mit HBAL		CPLEX mit HLOW	
satz	Zeit	Qual.	Zeit	Qual.	Zeit	Qual.	Zeit	Qual.
A	0.02	opt	0.03	0.22%	0.04	opt	0.19	0.00%
B	0.10	opt	0.28	0.43%	1.17	0.43%	1221.46	0.50%
C	5.62	0.00%	14.85	2.86%	4901.66	0.38%	?	?
D	40.22	0.02%	76.62	0.09%	34100.65	0.05%	?	?
E	167.11	0.03%	311.40	0.17%	10520.09	0.10%	?	?
F	62.68	0.00%	176.15	0.32%	22677.68	0.31%	?	?

Tabelle 4.15: Laufzeit und Lösungsqualität der exakten Homogenitätsmodelle

gleich sind zusätzlich noch unter der Überschrift „CPLEX ohne Hom." die Ergebnisse ohne Homogenitätskosten angegeben. Die Lösungsqualitäten der Varianten mit Homogenitätskosten (letzten drei Blöcke) beziehen sich für die Datensätze B bis F nur auf obere Schranken des optimalen Zielfunktionswerts, da die optimalen Zielfunktionswerte unbekannt sind. Man beachte, dass die Lösungen der Spalten „CPLEX ohne Hom." einer anderen Zielfunktionsbewertung unterliegen als die restlichen Spalten, da bei den „CPLEX ohne Hom."-Ergebnissen *keine* Homogenitätskosten berücksichtigt werden.

Die exakte Variante HLOW war nur für die beiden kleinsten Instanzen A und B in der Lage, Lösungen zu produzieren. Alle anderen Testläufe wurden nach einem Tag abgebrochen. Die HBAL-Variante schafft es immerhin, Lösungen für die ersten sechs Datensätze zu liefern. Die dabei benötigten Rechenzeiten sind allerdings enorm groß für die verhältnismäßig kleinen Testinstanzen. Die heuristische Variante unter der Überschrift „CPLEX mit Hom." ist bedeutend schneller und kommt in den meisten Fällen sehr nahe an die Lösungsqualität der exakten Verfahren heran. Eine Ausnahme bildet der Datensatz C, bei dem die heuristische Manipulation der Zielfunktion zu Lösungen führt, die mit 2.86% Abweichung vom Optimum verhältnismäßig schlecht ist.

Insgesamt die der Performance der exakten Modellierungen für Homogenitätskosten enttäuschend. Die IP-Modelle für die Flottenzuweisung werden dadurch sehr viel größer und schwerer zu lösen. Es konnte damit überhaupt nur für die sechs kleinsten Instanzen eine Lösung gefunden werden. Obwohl das Modell HLOW weniger Variablen und Gleichungen dem Grundmodell hinzufügt, ist seine Performance noch bedeutend schlechter als die des HBAL-Modells, da die LP-Relaxierung des HLOW-Modells praktisch unbrauchbar ist (siehe dazu Satz 4.16).

Heuristische Berücksichtigung Nachdem die exakten Ansätze in der Praxis nicht in der Lage sind, Lösungen in vernünftiger Zeit zu liefern, konzentrieren wir uns jetzt auf die heuristischen Ansätze zur Berücksichtigung von Homogenitätskosten. Die ohnehin heuristischen Lokale Suche Verfahren können diese Kosten einfach mitberechnen und in der Zielfunktion berücksichtigen, die IP-basierten Verfahren lösen zuerst die Instanz ohne die Berücksichtigung von Homogenitätskosten und modifizieren anhand dieses Ergebnisses nur die Zielfunktion, um Homogenitätskosten zu berücksichtigen (siehe Abschnitt 4.6.2.3).

Da wir für die Datensätze G bis O weder eine (gute) obere Schranke auf den Wert der optimalen Zielfunktion mit Homogenitätskosten geschweige denn den optimalen Wert selber

kennen, bewerten wir für diese Datensätze die Qualität von Lösungen mit Homogenitätskosten anhand der oberen Schranke des „CPLEX mit Hom.“- bzw. „CBC mit Hom.“-Laufs, je nachdem welche Schranke größer ist. Dabei handelt es sich aber *nicht* um eine obere Schranke für das tatsächliche Optimum, da die „CPLEX/CBC mit Hom.“-Läufe nur eine heuristische Zielfunktion verwenden, von der wir nur wissen, dass sie kleiner gleich der tatsächlichen Zielfunktion ist. Es ist daher zu erwarten, dass die in den Tabellen 4.16 und 4.17 angegebenen Gewinnabweichungen für die Datensätze G bis O zu gering sind.

Tabelle 4.16 zeigt die Laufzeiten und Lösungsqualitäten der Lokale Suche Verfahren mit und ohne Berücksichtigung von Homogenitätskosten. Die gleichen Angaben finden sich in Tabelle 4.17 für die IP-basierten Verfahren mit heuristischer Berücksichtigung von Homogenitätskosten.

Es fällt auf, dass vor allem HC an Lösungsqualität verliert, wenn Homogenitätskosten berücksichtigt werden müssen. Bei SA tritt dieser Effekt nicht ganz so deutliche auf, allerdings verliert auch SA gegenüber der normalen Variante zum Teil deutlich an Lösungsqualität, wenn Homogenitätskosten im Spiel sind (Datensatz C, G, K). Dabei muss ferner berücksichtigt werden, dass die Qualitätsangaben für die Datensätze G bis O eher noch zu optimistisch sind.

Die IP-basierten Verfahren schlagen sich deutlich besser, was die Lösungsqualität angeht. Sie sind den Lokale Suche Verfahren immer überlegen. Projiziert man die gesicherten Lösungsqualitäten für die Datensätze A bis F auf die restlichen Datensätze, kann man von einer Lösungsqualität von ungefähr 0.40% ausgehen, wobei einzelne Ausreißer zum Schlechten nicht ausgeschlossen sind, wie Datensatz C lehrt.

Die Laufzeiten der Lokale Suche Verfahren werden von den Homogenitätskosten nicht besonders verschlechtert. Teilweise kommt es zu einer Verdoppelung der Laufzeit (HC auf Datensatz D, SA auf Datensatz E), aber gerade das SA-Verfahren erfährt durch Homogenitätskosten auch deutlich kürzere Laufzeiten (Datensatz I und O). Da die Berechnung der Homogenitätskosten kaum zusätzliche Zeit erfordert, muss der Grund dafür bei der geänderten Zielfunktion gesucht werden. In manchen Fällen scheinen Homogenitätskosten sich förderlich auf die Konvergenz des adaptiven Simulated Annealing Algorithmus auszuwirken, in anderen Fällen stören sie die Konvergenz.

Die IP-basierten Verfahren müssen durch die heuristische Berücksichtigung von Homogenitätskosten doppelte Arbeit verrichten, da sie den Datensatz zuerst ohne Homogenitätskosten lösen müssen, um die Zielfunktion heuristisch anpassen zu können. Daher ist eine Verdoppelung der Laufzeit nicht überraschend. Es scheint ferner so zu sein, dass die angepasste Zielfunktion die Instanzen in vielen Fällen schwerer zu lösen macht, da in vielen Fällen die Laufzeiten um mehr als den erwarteten Faktor 2 anwachsen.

4.8.7 IP-basierte Verfahren: exakt vs. approximativ

Wir haben in den bisherigen Experimenten die prinzipiell exakt arbeitenden IP-basierten Verfahren nur approximativ verwendet, indem wir die Branch&Bound Suche beendet haben, sobald wir eine Lösung, die beweisbar nicht weiter als 0.5% vom Optimum entfernt ist, gefunden haben.

Daten-	HC ohne Hom.		HC mit Hom.		SA ohne Hom.		SA mit Hom.	
satz	Zeit	Qual.	Zeit	Qual.	Zeit	Qual.	Zeit	Qual.
A	0.00	opt	0.00	1.64%	0.04	opt	0.03	0.22%
B	0.02	0.52%	0.04	0.73%	0.19	0.01%	0.25	0.43%
C	3.18	1.27%	4.46	12.54%	47.39	0.74%	39.47	5.78%
D	4.61	0.87%	10.95	2.30%	41.59	0.25%	53.18	0.52%
E	6.62	1.22%	11.65	1.78%	98.19	0.31%	192.43	0.27%
F	7.13	0.58%	8.34	4.16%	119.07	0.33%	117.85	0.90%
G	8.52	0.90%	9.93	1.84%	136.83	0.14%	122.74	1.64%
H	8.06	2.40%	11.03	4.70%	73.55	1.31%	39.28	1.34%
I	17.59	1.26%	69.09	4.57%	438.93	0.18%	210.22	0.93%
J	13.89	1.08%	22.10	4.31%	140.25	0.39%	211.95	1.37%
K	5.58	0.26%	7.29	1.66%	284.10	0.08%	270.39	0.88%
L	0.37	0.52%	0.49	1.02%	2.62	0.16%	4.72	0.29%
M	9.62	3.12%	10.61	4.65%	70.55	0.94%	64.38	0.78%
N	19.63	2.82%	24.18	3.38%	161.70	1.70%	112.19	2.01%
O	6.94	2.86%	10.94	3.35%	328.49	1.18%	119.73	1.82%

Tabelle 4.16: Laufzeit und Lösungsqualität der Lokale Suche Verfahren mit und ohne Berücksichtigung von Homogenitätskosten

Daten-	CPLEX ohne Hom.		CPLEX mit Hom.		CBC ohne Hom.		CBC mit Hom.	
satz	Zeit	Qual.	Zeit	Qual.	Zeit	Qual.	Zeit	Qual.
A	0.02	opt	0.03	0.22%	0.01	opt	0.02	0.22%
B	0.10	opt	0.28	0.43%	0.07	opt	0.12	0.43%
C	5.62	0.00%	14.85	2.86%	15.34	0.00%	52.84	2.37%
D	40.22	0.02%	76.62	0.09%	61.86	0.01%	198.49	0.03%
E	167.11	0.03%	311.40	0.17%	154.40	0.00%	464.49	0.25%
F	62.68	0.00%	176.15	0.32%	100.19	0.00%	706.13	0.38%
G	74.16	0.01%	299.86	0.01%	86.08	0.00%	85.60	0.00%
H	38.05	0.00%	211.15	0.03%	60.86	0.00%	393.23	0.02%
I	285.16	0.02%	679.72	0.01%	657.17	0.02%	1030.43	0.01%
J	241.46	0.01%	665.74	0.02%	154.14	0.01%	729.21	0.03%
K	5.26	opt	19.38	0.00%	6.36	opt	36.62	0.00%
L	0.94	0.26%	3.38	0.09%	3.53	0.10%	8.42	0.10%
M	41.82	0.01%	170.49	0.07%	57.39	0.00%	135.01	0.05%
N	281.17	0.11%	741.57	0.05%	234.22	0.04%	347.58	0.08%
O	375.24	0.10%	809.01	0.09%	415.59	0.08%	325.68	0.05%

Tabelle 4.17: Laufzeit und Lösungsqualität der IP-basierten Verfahren mit und ohne heuristischer Berücksichtigung von Homogenitätskosten

Daten-	exakt		approximativ		
satz	Knoten	Zeit	Knoten	Zeit	Fehler
A	0	0.01	0	0.01	-
B	0	0.07	0	0.07	-
C	28	6.79	7	5.62	0.00%
D	182340	25150.17	98	40.22	0.02%
E	2006	910.18	106	167.11	0.03%
F	5408	1572.99	21	62.68	0.00%
G	427	254.23	21	74.16	0.01%
H	36	45.44	6	38.05	0.00%
I	109534	101344.99	50	285.16	0.02%
J	20064	8680.36	100	241.46	0.01%
K	0	5.26	0	5.26	-
L	44405	253.21	20	0.94	0.26%
M	967472	137096.96	82	41.82	0.01%

Tabelle 4.18: Laufzeiten und besuchte Branch&Bound-Knoten von CPLEX im exakten bzw. approximativen Modus bei maximalem Preprocessing

Die dabei tatsächliche auftretenden so genannten *integrality gaps* sind zumeist erheblich kleiner und liegen im Bereich von 0.02%. Trotzdem ist das exakte Lösen dieser Probleme extrem aufwendig.

Um dies zu zeigen, haben wir in Tabelle 4.18 die Datensätze A bis M einmal exakt bis zum beweisbaren Optimum und einmal wie bisher approximativ mit CPLEX gelöst. CPLEX ist gegenüber CBC für das exakte Lösen besser geeignet, da hier anscheinend stärkere Cuts generiert werden. Neben den Laufzeiten in Sekunden geben wir in den Spalten „Knoten“ die Anzahl besuchter Branch&Bound Knoten an und zeigen in der Spalte „Fehler“ den relativen Gewinnverlust der approximativen gegenüber der optimalen Lösung an.

Die auftretende Laufzeitverlängerung beim Wechsel vom approximativen zum exakten Lösen ist teilweise gigantisch. Bei Instanz M wächst die Laufzeit um einen Faktor von über 3000 auf über 38 Stunden. Dabei müssen fast 12000 mal so viele Knoten während der Branch&Bound Suche betrachtet werden. Ähnlich schlimm verhält es sich mit den Datensätzen D, I und L. Aber auch bei den meisten anderen Datensätzen steigt die Laufzeit um Faktoren an, wenn man nach einer optimalen Lösung sucht. Tendenziell kann man sagen, dass je tiefer im Branch&Bound Baum die erste ganzzahlige Lösung gefunden wird, desto länger benötigt man zum Berechnen des Optimums.

Die Experimente in diesem Abschnitt sind dabei mit den vollständig aufbereiteten Eingabeinstanzen durchgeführt worden. Ohne Preprocessing wären die Laufzeiten noch sehr viel höher. Aber selbst bei maximalem Preprocessing ist es nicht gelungen, die größten Instanzen N bis Q optimal zu lösen. Die entsprechenden Läufe sind nach drei Tagen ohne Ergebnis abgebrochen worden.

Daten-	SA		CBC	
satz	zyklisch	azyklisch	zyklisch	azyklisch
A	0.04	0.05	0.01	0.02
B	0.19	0.29	0.07	0.12
C	47.39	52.24	15.34	20.13
D	41.59	49.58	61.86	48.63
E	98.19	107.92	154.40	145.45
F	119.07	151.90	100.19	110.08
G	136.83	202.59	86.08	112.07
H	73.55	135.80	60.86	76.19
I	438.93	397.90	657.17	435.87
J	140.25	268.08	154.14	218.88
K	284.10	305.22	6.36	7.80
L	2.62	17.90	3.53	1.40
M	70.55	234.65	57.39	64.53
N	161.70	274.66	234.22	260.22
O	328.49	238.07	415.59	204.11

Tabelle 4.19: Laufzeiten der Verfahren SA und CBC auf zyklischen und azyklischen Flottenzuweisungsinstanzen bei maximalem Preprocessing

4.8.8 Zyklische vs. azyklische Flottenzuweisungsprobleme

Zum Schluss wollen wir noch kurz untersuchen, ob sich zyklische oder azyklischen Instanzen schneller lösen lassen. Dazu haben wir die zyklischen Instanzen A bis O sowohl mit einem zyklischen als auch einem azyklischen Flottenzuweisungsoptimierer gelöst. Zyklische Datensätze eignen sich als Eingabe für beide Varianten von Flottenzuweisungsproblemen und lassen so einen direkten Vergleich der Leistungsfähigkeit der zyklischen bzw. azyklischen Löser zu. Da jede zyklische Lösung eines Flottenzuweisungsproblems insbesondere auch eine Lösung für die entsprechende azyklische Variante darstellt, haben die azyklischen Optimierer mehr Freiheiten als ihre zyklischen Kollegen.

In Tabelle 4.19 sind die Ergebnisse für die beiden Lösungsverfahren SA und CBC dargestellt. Wir beschränken uns hier auf diese beiden Verfahren, die stellvertretend für die Lokale Suche und IP-basierten Verfahren stehen. Es wird für jeden Datensatz die Laufzeit in Sekunden angegeben, die der zyklische bzw. azyklische Optimierer zum Lösen benötigt.

Die Unterschiede zwischen zyklischen und azyklischen Instanzen sind bei beiden Verfahren nicht besonders groß. CBC löst mal die zyklische und mal die azyklische Instanz schneller. Ein klarer Trend ist hier nicht zu erkennen. Bei den IP-basierten Verfahren hängt es sehr stark davon ab, wie früh man während der Branch&Bound Suche auf die erste ganzzahlige Lösung stößt, und dies lässt sich im Einzelfall kaum vorhersagen.

Für SA gilt ähnliches. Mal wird die zyklische, mal die azyklische Instanz schneller gelöst. Allerdings gewinnt der azyklische SA-Löser nur bei den beiden Instanzen I und O. Die anderen Male hat der zyklische SA-Löser die Nase vorn. Das mag ein Indiz dafür sein, dass das

azyklische Flottenzuweisungsproblem von unseren Lokale Suche Verfahren ein wenig langsamer gelöst werden kann als das zyklische Flottenzuweisungsproblem. Die Freiheitsgrade sind im azyklischen Fall ein wenig größer, was sich auch in einer etwas größeren Nachbarschaft wiederspiegelt.

4.9 Zusammenfassung

Wir haben in diesem Kapitel verschiedene neue heuristische und exakte Optimierungsverfahren für das Flottenzuweisungsproblem in der Flugplanung vorgestellt.

Die heuristischen Lösungsverfahren basieren auf der Lokale Suche-Idee und verwenden eine problemspezifische Nachbarschaft, die speziell für das Flottenzuweisungsproblem entwickelt worden ist und die schnell (in deutlich weniger als einer Millisekunde) *zulässige* Nachbarn zu einer aktuellen Lösung produzieren kann. Die erreichbare Lösungsqualität ist für das Simulated Annealing Verfahren mit durchschnittlich 0.5% gut. Das Hill Climbing Verfahren fällt demgegenüber etwas ab, hat aber den Vorteil der mit Abstand schnellste Flottenzuweisungsoptimierer zu sein, so dass er sich zur schnellen Beurteilung von alternativen Planungsszenarien eignet.

Die Nachbarschaft der Lokale Suche Verfahren lässt sich leicht an erweiterte Anforderungen für das Flottenzuweisungsproblem wie beispielsweise verbindungsabhängige Mindestbodenzeiten oder die Berücksichtigung von Homogenitätskosten anpassen, ohne dabei allzuviel an Performance zu verlieren. Insbesondere können die Lokale Suche Verfahren durch die Integration mit den Planungsprozessen Marktmodellierung und Ertragsmanagement (Kapitel 6) eine verbesserte und genauere Lösungsbewertung bei der Optimierung einsetzen, und damit die *reale* Qualität der produzierten Lösungen nochmals deutlich verbessern.

Die exakten Verfahren sind IP-basiert und bauen auf dem Time Space Network Modell für die Flottenzuweisung auf. Die Lösungszeiten der Verfahren liegen dabei im gleichen Bereich wie die Optimierungszeiten des heuristischen Simulated Annealing Verfahrens, wenn man die Eingabeinstanzen durch den Einsatz von Preprocessing-Techniken vor der eigentlichen Optimierung heuristisch verkleinert und darauf verzichtet, die Branch&Bound Suche bis zum Ende auszuführen, und nach dem Finden einer Lösung mit beweisbar guter Qualität die Suche abbricht. Die durch das heuristische Preprocessing verursachten Gewinneinbußen liegen dabei mit durchschnittlich 0.3% in einem akzeptablen Rahmen. Die aus dem nur approximativen Einsatz der Branch&Bound Suche resultierende Abnahme an Lösungsqualität ist dagegen mit zumeist deutlich weniger als 0.1% vernachlässigbar. Damit schlagen sie die Simulated Annealing Heuristik, da sie in vergleichbarer, teilweise sogar kürzerer Zeit qualitativ bessere Lösungen liefern.

Wir konnten auch die exakten Verfahren um die Fähigkeit erweitern, verbindungsabhängige Mindestbodenzeiten (und Gewinne) und Homogenitätskosten zu unterstützen. Verbindungsabhängige Mindestbodenzeiten sind dabei exakt modelliert worden und führen auf realen Testinstanzen zu einer um den Faktor 2 bis 3 höheren Laufzeit. Wir konnten auch die Berücksichtigung von Homogenitätskosten in der Zielfunktion exakt modellieren, allerdings sind die resultierenden IP-Modell wegen ihrer Größe und damit verbundenen extrem langen Lösungszeit kaum brauchbar. Eine heuristische Umsetzung von Homogenitätskosten konnte

dagegen überzeugen und liefert qualitativ hochwertige Ergebnisse bei ungefähr verdoppelter Laufzeit gegenüber dem Grundmodell.

Die von den exakten Verfahren aufgestellten IP-Modelle sind mit Standard-IP-Optimieren gelöst worden. Dabei kam zum einen das kommerzielle ILOG CPLEX und zum anderen das freie CBC aus der COIN-OR-Library zum Einsatz. Im direkten Vergleich konnte sich dabei CPLEX praktisch kaum von CBC absetzen.

Abschließend kann gesagt werden, dass sich mit den in diesem Kapitel präsentierten Verfahren auch große reale Flottenzuweisungsprobleme mit mehr als 40000 Legs fast optimal in weniger als einer Stunde lösen lassen. Die Lokale Suche Verfahren haben dabei ihre volle Praxistauglichkeit bei mehreren internationalen Fluggesellschaften seit einigen Jahren unter Beweis gestellt und geholfen, die Flotteneinsatzplanung entscheidend zu verbessern.

Damit werden natürlich auch neue Begehrlichkeiten auf Seiten der Fluggesellschaften geweckt. Der Trend geht hier zunehmend in Richtung einer stärkeren Integration der verschiedenen Planungsprozesse. Zusätzliche Anforderungen, die ihren Ursprung in benachbarten Planungsprozessen haben, sollen daher bereits bei der Flottenzuweisung berücksichtigt werden. Dazu gehören

- die Verwendung einer genaueren Zielfunktion für die Bewertung von Flottenzuweisungen,
- das Erstellen von robusten Plänen, die unempfindlicher gegenüber Störungen sind,
- die Berücksichtigung von Wartungsaspekten,
- die Integration mit der Crewplanung,
- usw.

Auf die ersten beiden Punkte gehen wir in den folgenden beiden Kapiteln noch näher ein.

— 5 –

Das stochastische Flottenzuweisungsproblem

In diesem Kapitel führen wir das stochastische Flottenzuweisungsproblem ein, bei dem manche der Eingabedaten, wie beispielsweise die Blockzeit, keine feste Zahl sondern eine Zufallsvariable sind, mit der sich Störungen im Flugbetrieb modellieren lassen. Ziel ist es hier darauf zu achten, dass die produzierten Flottenzuweisungen unempfindlich gegenüber Störungen sind und gegebenenfalls einfach repariert werden können.

Dabei handelt es sich nicht um eine künstliche Erweiterung des Flottenzuweisungsproblems sondern um ein bedeutendes Real-World-Problem, mit dem Fluggesellschaften täglich im Störungsmanagement konfrontiert sind und das trotzdem bisher nicht untersucht worden ist.

Neben der Erkenntnis, dass es sich bei der stochastischen Flottenzuweisung um ein PSPACE-vollständiges Problem handelt, präsentieren wir in diesem Kapitel eine generische Methode, wie sich das stochastische Flottenzuweisungsproblem als ein spezielles 2-Personen-Spiel modellieren und über eine Spielbaumauswertung lösen lässt.

5.1 Motivation

Ein Plan ist eine Lösung eines Zuweisungsproblems, das sich über mehrere Zeitschritte erstreckt. Der größte Teil der Forschung im Bereich der Optimierung von Plänen beschäftigt sich mit der Generierung statischer, vorberechneter Pläne in dem Sinne, dass man annimmt, die zum Planungszeitpunkt vorliegenden Plandaten seien vollständig bekannt und unveränderbar. Traditionellerweise werden Pläne erzeugt, die das Ziel haben, den Gewinn unter Verwendung von festen Schätzdaten bzw. „Erwartungsdaten“ zu maximieren.

Sobald ein vorberechneter Plan in der echten Welt eingesetzt wird, werden verschiedene Unsicherheitsaspekte des Systems, in dem der Plan abläuft, zu Störungen führen, so dass der ursprüngliche Plan seine Güte oder sogar seine Zulässigkeit verliert. Es ist dann die Aufgabe des Störungsmanagements, einen teilweise neuen Plan zu entwickeln, der das System wieder in einen zulässigen Zustand überführt. Der alte Plan wird also schnell außer Kraft gesetzt, bzw. muss „repariert“ werden, oder es muss sogar komplett neu geplant werden. Dies verursacht im Allgemeinen beträchtliche Kosten, die den eigentlich vorausgeplanten Gewinn massiv schmälern können.

Nichtsdestotrotz wird häufig ein vorberechneter Plan benötigt, um sicherzustellen, dass die vorhandenen Ressourcen prinzipiell ausreichen, um langfristige Ziele optimieren zu können und um als Grundlage für nachgelagerte Planungsstufen zu dienen. Im Falle von Fluggesellschaften, aber auch anderen fahrplangebundenen Verkehrsbetrieben wie ÖPNV oder Eisenbahngesellschaften, dienen Pläne ferner als Kommunikationswerkzeug mit den Kunden und stellen damit das für den Kunden sichtbare Produkt des Unternehmens dar.

Erstrebenswert ist es allerdings, dass der Plan so gebaut wird, dass er sich im Falle von auftretenden Störungen einfach, schnell und kostengünstig reparieren lässt. Auch die vom Störungsmanagement ausgearbeiteten Neu- oder Umplanungen sollten diese Eigenschaft besitzen, da anschließend mit weiteren Störungen zu rechnen ist. Insofern lassen sich die beiden Probleme, das initiale Erstellen eines Plans und das Beheben von Störungen, nicht völlig voneinander trennen.

Häufig besitzen Planer Informationen über das reale Verhalten einer sich verändernden Umgebung in Form von Wahrscheinlichkeitsverteilungen über die Eingabedaten, ignorieren diese aber bewusst, um die Planung, die auch so schon kompliziert und aufwendig genug ist, zu vereinfachen.

Wir stellen in diesem Kapitel das *Reparaturspiel* vor, eine generische Methodik für allgemeine mehrstufige Planungsprobleme, die mit stochastischen Eingabegrößen sinnvoll umgehen kann, indem sie beispielsweise die *erwarteten Gewinne* optimiert. Das Verfahren basiert auf der Modellierung der Planungsaufgabe als spezielles 2-Personen-Spiel und der Auswertung des resultierenden Spielbaums mittels eines modifizierten Alphabeta-Algorithmus.

5.1.1 Szenario für die stochastische Flottenzuweisung

Wir betrachten hier hauptsächlich das Problem der stochastischen Flottenzuweisung im Bereich des Störungsmanagements. Gegeben ist ein zulässiger azyklischer Flugplan inklusive Flottenzuweisung, der umgesetzt werden soll. Im Störungsmanagement, das Teil des Operations Control ist, haben wir es im Gegensatz zur langfristigen Planung immer mit dem Flugplan eines konkreten Zeitraums zu tun, so dass hier nur die azyklische Variante des Flottenzuweisungsproblems Sinn macht. Während der operationellen Durchführung treten nun unvorhergesehene Ereignisse (Störungen) auf, die die vorausberechnete Planung unzulässig machen können. Wir betrachten hier zwei Arten von Störungen:

- Ein Leg weist eine gegenüber der ursprünglichen Planung verlängerte Blockzeit auf. Das verspätete Flugzeug kann also unter Umständen nicht das ihm zugedachte Folgeleg erreichen.

- Ein Leg fällt komplett aus, entweder wegen eines schwerwiegenden technischen Defekts oder als Folge monetärer Überlegungen, da das Leg nicht gut ausgelastet ist.

Aufgabe des Störungsmanagements ist es nun, bei Bekanntwerden von Störungen unverzüglich zu reagieren und den Flugplan, wenn nötig, zu reparieren. Dabei können vom Störungsmanagement folgende Aktionen veranlasst werden:

- Verschieben des Starts eines Legs nach hinten. Kurze Verspätungen können so abgefangen werden. Es besteht aber die Gefahr, dass sich dadurch Verspätungen weit in die Zukunft fortpflanzen.
- Ändern der Flottenzuweisung eines Legs. Unter Umständen steht von einer anderen Flotte ein Flugzeug zur Verfügung, das einen wegen einer Verspätung verpassten Anschlussflug übernehmen kann.
- Streichen eines Legs. Gerade bei Ausfall eines Legs ist das Streichen weiterer Legs oft die einzige Möglichkeit, wieder einen zulässigen Flugplan zu produzieren.

Durch diese Aktionen soll wieder ein zulässiger Flugplan erzeugt werden. Dabei kann natürlich nur Einfluss auf zukünftige, noch nicht gestartete Legs genommen werden. Das Hauptanliegen ist dabei, so wenig Änderungen wie möglich am Ursprungsplan vorzunehmen. Gründe dafür sind:

- Der Ursprungsplan ist für den Kunden das Produkt der Fluggesellschaft und Änderungen sorgen hier schnell für unzufriedene Kunden.
- Der Ursprungsplan ist unter der Prämisse der Gewinnmaximierung erstellt worden, und große Änderungen haben im Allgemeinen negative Auswirkungen auf den erzielbaren Gewinn.
- Der Ursprungsplan dient als Grundlage für eine ganze Reihe weiterer Planungen. Crews müssen den Legs zugewiesen werden und übernehmen häufig mehrere Legs, die ein Flugzeug nacheinander bedient, so dass Änderungen von Folgelegs gravierende Auswirkungen im Crew-Bereich nach sich ziehen. Auch die Bodenabfertigung, Wartung, usw. sind auf verlässliche Informationen über den Flugplan mit seiner Flottenzuweisung angewiesen.

Die Qualität einer Umplanung wird daher anhand ihrer getätigten Änderungen bewertet. Zusammen mit unserem Industriepartner, Lufthansa Systems, haben wir für jede der oben aufgeführten Änderungsaktionen Kosten definiert, die die Schwere der Änderung ausdrücken. Legstreichungen sind am teuersten, danach kommen Flottenwechsel. Das Verschieben von Legs wird in Abhängigkeit von der Dauer der Verschiebung bewertet. Dazu kommen noch Änderungen auf der Einnahmeseite, da durch Flugplanänderungen unter Umständen Passagiere zu anderen Fluggesellschaften wechseln. Es kann aber in seltenen Fällen auch der Fall eintreten, dass durch Änderungen zusätzliche Erlöse erzielt werden können, da sich neue Umsteigemöglichkeiten ergeben oder durch einen Flottentausch zusätzliche Sitzplätze verfügbar werden.

Bis hierher lässt sich das Problem der Flottenzuweisung im Störungsmanagement als eine deterministische Planungsaufgabe ansehen. In der Praxis wird sie bis heute auch als solche behandelt. Wir stellen in Abschnitt 5.4.2.2 ein auf dem Time Space Network aufbauendes Modell vor, das diese deterministische Planungsaufgabe löst.

Nun ist aber das Störungsmanagement ein fortlaufender Prozess. Nachdem eine Störung behoben worden ist, tritt eher früher als später die nächste Störung auf, die eine Umplanung erforderlich macht. Wir gehen hier davon aus, dass zukünftige Störungen nicht vollkommen unerwartet eintreten, sondern das zu jedem Leg eine diskrete Wahrscheinlichkeitsverteilung gegeben ist, die mögliche Verspätungen und Ausfälle beschreibt. Dann lässt sich das Störungsmanagement als ein mehrstufiger Entscheidungsprozess unter Unsicherheit ansehen - der Optimierer repariert einen gestörten Plan, anschließend treten zufällig neue Störungen ein, woraufhin der Optimierer wieder in Aktion treten muss, usw. - und die Bewertung einer Umplanung für eine akute Störung sollte nicht nur von der aktuellen Situation sondern auch von möglichen zukünftigen Störungen abhängig gemacht werden. Das heißt, neben den tatsächlich anfallenden Kosten für eine durchzuführende Umplanung sollten auch die *erwarteten* Kosten für zukünftige Umplanungen mitberücksichtigt werden, und wir erhalten *das stochastische Flottenzuweisungsproblem im Störungsmanagement*.

Neben der Verwendung im Störungsmanagement lässt sich die hier beschriebene stochastische Flottenzuweisung allerdings auch beim Erstellen der initialen Flottenzuweisung einsetzen:

- Es lassen sich Zeitpunkte identifizieren, die hohe erwartete Umplanungskosten verursachen. Dafür simuliert man die während der Planungsperiode möglicherweise auftretenden Störungen und protokolliert die anfallenden Umplanungskosten.
- Sind erst einmal die kritischen Zeitpunkte eines Flugplans identifiziert, kann man versuchen, durch lokale Neuplanungen die Robustheit des Planes zu erhöhen und damit die erwarteten Umplanungskosten zu senken.

5.1.2 Literaturübersicht

Der Bereich der Optimierung von stochastischen Flottenzuweisungsproblemen ist bislang in der Literatur nicht betrachtet worden, obwohl das Berücksichtigen von stochastischen Eingabedaten als einer der bedeutenden und gewinnsteigernden Aspekte in der Flugplanung angesehen wird [Barnhart et al., 2003].

Die hier betrachteten Planungsaufgaben fallen in den Bereich der „Mehrstufigen Entscheidungen unter Unsicherheit“. Es ist ein Teilgebiet des größeren Feldes von entscheidungstheoretischen Ansätzen [Horvitz et al., 1988], zu dem auch die (lineare) stochastische Programmierung gehört [Engell et al., 2001], [Römisch and Schultz, 2001].

Manchmal wird in diesem Zusammenhang der Begriff „robuster Plan“ verwendet, der in zweierlei Bedeutung auftritt: Zum einen informell als Plan, dessen Wert relativ unsensibel auf mögliche Realisierungen der echten Welt reagiert, aber dann auch als Plan, der bei schlimmstmöglichem Verlauf der exogenen Einflüsse noch gültig bleibt. Bei robuster Planung (vgl.

[Scholl, 2001]) geht man also von einer grundsätzlichen Risikoscheu des Anwenders bei Entscheidungssituationen mit ausgeprägter Unsicherheit der verfügbaren Informationen aus. Wir verstehen in diesem Kapitel unter „robust“ die informelle Interpretation, sehen unsere Arbeiten aber im Bereich zwischen robuster Optimierung und stochastischer Optimierung, wobei wir uns zunächst am Erwartungswert orientieren und nicht an Minmax-, (μ, σ)-, Bernoulli- oder anderen Kriterien [Daniels and Kouvelis, 1995], [Kouvelis et al., 2000], [Mulvey et al., 1995, Scholl, 2001].

Die Erkenntnis, dass sich mehrstufige Entscheidungsprobleme unter Unsicherheit in Form von speziellen 2-Personen-Spielen ausdrücken lassen, stammt bereits von [Papadimitriou, 1985]. Er geht in seiner Arbeit allerdings nur auf die theoretischen Aspekte ein und zeigt, dass die Klasse der mehrstufigen Entscheidungsprobleme unter Unsicherheit (unter moderaten, eher technischen Einschränkungen) der Klasse PSPACE entspricht. [Leon et al., 1994] stellen in ihrer Arbeit ein Verfahren vor, das diese Erkenntnis speziell für ein Job-Shop-Problem umsetzt. Die Algorithmus ist allerdings stark abhängig von dem betrachteten Job-Shop-Problem, und die resultierenden Spielbaumauswertung ist nicht besonders effizient.

Der Alphabeta-Algorithmus [Knuth and Moore, 1975] bildet die Grundlage der Spielbaumsuchalgorithmen. In professionellen Computerspiel-Programmen [Donninger et al., 2004] wird meistens die Negascout-Variante [Reinefeld, 1983] des Alphabeta-Algorithmus benutzt. Die wichtigste Beobachtung über die letzten 40 Jahre hinweg in Spielen wie Schach und Ähnlichen ist dabei, dass der Spielbaum wie ein Fehlerfilter wirkt. Deshalb gilt: „Je schneller und je intelligenter der Suchalgorithmus ist, desto besser werden die Spielergebnisse.“ Dass dieses nicht selbstverständlich ist, zeigen theoretische Untersuchungen zu Fehlerfortpflanzungen von [Nau, 1979], [Althöfer, 1988] und [Lorenz and Monien, 2004].

Es ist nicht für jedes mehrstufige Entscheidungsproblem unter Unsicherheit sinnvoll zu versuchen, es mittels aufwendiger Verfahren mit potentiell exponentieller Laufzeit wie der Spielbaumsuche zu lösen. In manchen Fällen lassen sich schnelle Approximationsverfahren mit polynomieller Laufzeit angeben, so zum Beispiel für ein in [Shmoys and Swamy, 2004] beschriebenes stochastisches Set-Cover Problem oder für ausgewählte stochastische Scheduling Probleme [Mörhing et al., 1999]. Allerdings handelt es sich bei der stochastischen Flottenzuweisung um ein PSPACE-vollständiges Problem (Abschnitt 5.3), dessen deterministische Variante bereits NP-vollständig und nicht approximierbar ist (Kapitel 3). Hier ist nicht zu erwarten, dass sich polynomielle Approximationsalgorithmen finden lassen.

5.2 Games against Nature

Papadimitriou führt in seinem Artikel [Papadimitriou, 1985] einen neuen Formalismus für Entscheidungsprobleme unter Unsicherheit ein, der auf 2-Personen-Spielen mit einem zufällig agierenden Gegenspieler beruht. Er nennt diese Spiele *Games against Nature* und zeigt, dass es sich dabei (unter sehr allgemeinen Annahmen) um eine alternative Charakterisierung der Klasse PSPACE handelt.

Definition 5.1 (Game against Nature). *Eine Problemklasse Π ist ein* Spiel gegen die Natur *(Game against Nature), wenn für Π und jede Instanz x von Π (mit Eingabelänge $|x|$) gilt:*

- *Es existieren ein festes Polynom p und eine feste Konstante $d \in \mathbb{N}$ für Π.*
- *Zu x existieren Mengen von Zuständen $Z_0, \ldots, Z_{p(|x|)}$. Jeder Zustand $s \in Z_i$ lässt sich in Platz $O(p(|x|)$ kodieren. Z_i enthält die möglichen Zustände nach der i-ten Entscheidung. Z_0 enthält nur den* initialen Zustand *und $Z_{p(|x|)}$ die* Endzustände.
- *Jeder Nicht-Endzustand besitzt d viele, mit ihm assoziierte Entscheidungen $\{1, \ldots, d\}$.*
- *Zu jeder Entscheidung j eines Zustands $s \in Z_i$ existiert eine diskrete Wahrscheinlichkeitverteilung über Z_{i+1}, die wir mit $D_s^j = \{(s_1^j, w_1^j), \ldots, (s_m^j, w_m^j)\}$ bezeichnen. Dabei gehören die s_k^j zur Zustandsmenge Z_{i+1} und die Wahrscheinlichkeiten w_k^j sind $p(|x|)$-bittige rationale, echt-positive Zahlen mit $\sum_{k=1}^{m} w_k^j = 1$. Ferner darf D_s^j nur maximal $p(|x|)$ viele Wahrscheinlichkeiten echt größer Null enthalten, das heißt es muss $m \leq p(|x|)$ gelten.*
- *Zu jedem Endzustand s sind ganzzahlige Kosten $c(s)$ definiert, wobei $|c(s)| \leq 2^{p(|x|)}$ ist.*
- *Zu Π gehören zwei feste Algorithmen A und B mit Laufzeit $O(p(n))$.*
  - *Bei Eingabe eines Instanz x, eines Zustands s, einer Entscheidung j und einer Zahl k liefert A das Paar (s_k^j, w_k^j) zurück.*
  - *Bei Eingabe einer Instanz x und eines Zustands s entscheidet B, ob s ein Endzustand ist, und gibt für Endzustände zusätzlich $c(s)$ zurück.*

Die Aufgabe ist es, eine beste Strategie (eine Strategie mit minimalen bzw. maximalen erwarteten Kosten) für eine Instanz x zu ermitteln, bzw. (für das zu Π korrespondierende Entscheidungsproblem) zu entscheiden, ob es eine Strategie mit Kosten kleiner gleich (Minimierungsproblem) bzw. größer gleich (Maximierungsproblem) einer vorgegebenen Schranke gibt.

Dabei ist noch zu klären, was wir unter einer Strategie und deren erwartete Kosten verstehen wollen. Wir definieren dazu:

Definition 5.2 (Spielbaum). *Der Spielbaum T der Instanz x eines Spiels gegen die Natur ist wie folge definiert:*

- *Der Baum besteht aus $2p(|x|) - 1$ Ebenen $Z_0, E_0, Z_1, E_1, \ldots, E_{p(|x|)-1}, Z_{p(|x|)}$. Die Knoten in Ebene Z_i entsprechen den Zuständen Z_i von x und die Knoten der Ebene E_i entsprechen allen Entscheidungen der Zustände aus Z_i.*
- *$s \in Z_i$ ist mit $j \in E_i$ verbunden, wenn j eine der d Entscheidungen von s ist. Die entsprechenden Kanten heißen* Entscheidungskanten.

  Ist j die j-te Entscheidung von s, so verlaufen von j Kanten zu allen Zuständen in der Wahrscheinlichkeitsverteilung $D_s^j = \{(s_1^j, w_1^j), \ldots, (s_m^j, w_m^j)\}$. Die Kante zum Zustand s_k^j wird mit der Wahrscheinlichkeit w_k^j markiert und Zufallskante *genannt.*
- *Die Knoten s der letzten Ebene $Z_{p(|x|)}$ werden mit ihren Kosten $c(s)$ markiert.*

Streng genommen muss es sich bei dem Spielbaum nur um einen Level-Graphen und nicht um einen Baum handeln, da verschiedene Entscheidungen bzw. Zufallsereignisse später wieder zu ein und demselben Folgezustand führen dürfen. Wir bleiben aber bei der Bezeichnung „Spielbaum“, da sie auch in diesem Fall gebräuchlicher ist.

Wir assoziieren zwei Spieler mit den Knoten eines Spielbaums. Zu den Zustandsknoten gehört der Spieler „Optimierer“, der für einen gegebenen Zustand die nächste (möglichst optimale) Entscheidung zu treffen hat. Zu den Entscheidungsknoten gehört der Spieler „Natur“, der zufällige Ereignisse auslösen kann. Wir nennen daher Zustandsknoten auch Optimiererknoten und Entscheidungsknoten auch Naturknoten.

Definition 5.3 (Strategie). *Eine Strategie S eines Spielbaums T eines Spiels gegen die Natur ist ein Teilgraph von T, den man erhält, wenn für jeden Zustandsknoten $s \in Z_i$ alle bis auf eine ausgehende Entscheidungskante löscht. Der zusammenhängende Graph, der den initialen Zustand (die Wurzel des Baumes) enthält, ist dann eine Strategie von T.*

Mit einer Strategie wird damit dem Optimierer ein „Regelwerk“ in die Hand gegeben, wie er sich in jeder Situation beim eintreten eines Zufallsereignisses als nächstes verhalten soll.

Definition 5.4 (Kosten einer Strategie/eines Spielbaums). *Die (erwarteten) Kosten einer Strategie S sind rekursiv definiert, indem jedem Knoten v der Strategie Kosten $c(v)$ zugeordnet werden:*

- *Für Endzustände $s \in Z_{p(|x|)}$ entsprechen die Kosten in der Strategie den gegebenen Kosten $c(v)$ im Spiel gegen die Natur.*
- *Für innere Zustandsknoten $s \in Z_i$ sind die Kosten gleich den Kosten des einzigen Kindes (einem Entscheidungsknoten) von s.*
- *Für einen Entscheidungsknoten $j \in E_i$, der die j-te Entscheidung von Zustand $s \in S_i$ ist und zu dem die diskrete Wahrscheinlichkeitsverteilung $D_s^j = \{(s_1^j, w_1^j), \ldots, (s_m^j, w_m^j)\}$ gehört, gilt:*

  $$c(j) = \sum_{k=1}^{m} w_k^j \cdot c(s_k^j)$$

  Hier werden also die erwarteten *Kosten der Entscheidung bestimmt.*
- *Die Kosten $c(S)$ der Strategie sind dann die Kosten des initialen Zustands.*

Eine Strategie S mit minimalen Kosten wird optimale *Strategie genannt.*[1] *Die Kosten $c(T)$ eines Spielbaums T sind die Kosten einer optimalen Strategie.*

Um eine optimale Strategie und die Kosten eines Suchbaums T zu bestimmen, müssen nicht alle möglichen Strategien von T bewertet werden. Ein Suchbaum lässt sich auch direkt bewerten und daraus eine optimale Strategie ablesen. Dazu seien die Kosten eines Knotens

[1] Im Falle eines Maximierungsproblems ist eine Strategie mit maximalen Kosten optimal.

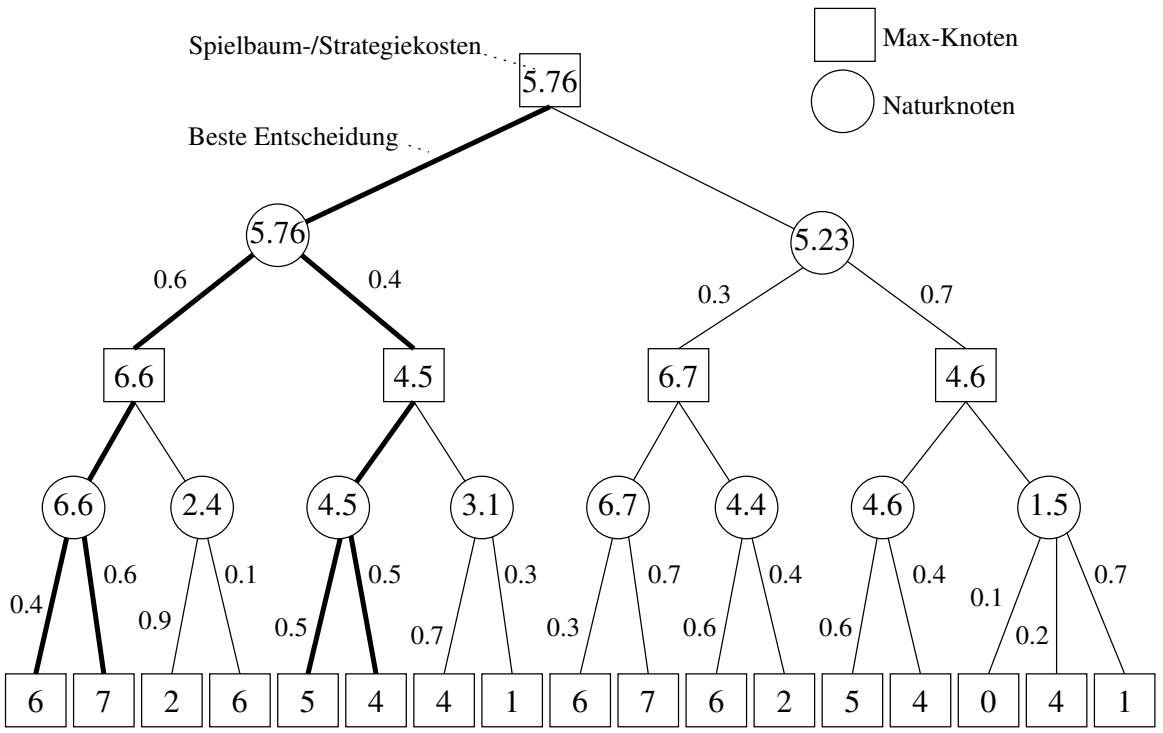


Abbildung 5.1: Ein Spielbaum mit einer optimalen (maximalen) Strategie

im Suchbaum für Endzustände und Entscheidungsknoten wie im Fall von Strategien definiert. Für innere Zustandsknoten s mit Kindern (Entscheidungen) $\{1, \ldots, d\}$ sei:

$$c(s) = \min_{j=1}^{d} c(j)$$

Die Kosten der Wurzel von T (initialer Zustand) entsprechen dann den Kosten des Baumes und man erhält eine optimale Strategie, indem man für jeden inneren Zustandsknoten s eine Kante zu einem Entscheidungskind mit minimalen Kosten auswählt.[2]

In Abbildung 5.1 ist ein Spielbaum inklusive der Kostenwerte der einzelnen Knoten dargestellt. An den Ausgangskanten der Naturknoten stehen die Wahrscheinlichkeiten, mit denen die entsprechenden Folgezustände erreicht werden. Die fetten Linien bilden die optimale (maximale) Strategie des Spielbaums.

Praktisch jedes Entscheidungsproblem unter Unsicherheit *mit einer polynomiell beschränkten Anzahl an aufeinanderfolgenden Entscheidungen* lässt sich als Spiel gegen die Natur auffassen. Die Einschränkungen in Definition 5.1 sind eher technischer Natur:

- Die Einschränkung, dass in jedem Zustand nur konstant viele Entscheidungen möglich sind, lässt sich durch das Einfügen von polynomiell vielen Zwischenzuständen auf eine exponentielle Anzahl an Entscheidungen je Zustand anheben. Wichtig ist, dass dabei die Anzahl an Zustandsmengen Z_i polynomiell bleibt.

[2] Im Falle von Maximierungsproblemen bildet man das Maximum über die Kosten der Entscheidungskinder.

- Entsprechendes gilt für die Einschränkung, dass die Zufallsverteilungen nur polynomiell viele echt positive Wahrscheinlichkeiten besitzen dürfen.
- Häufig werden bei Entscheidungsproblemen die Kosten nicht von den Endzuständen bestimmt, sondern von den Entscheidungen und Zufallsereignissen, die zu einem Endzustand führen. Da die Zustände ausreichend Platz bieten, ist es hier möglich, sich in den Zuständen die Historie an Entscheidungen und Zufallsereignissen zu merken und diese Informationen im Endzustand zur Berechnung der Kosten zu verwenden.

Papadimitriou hat nun gezeigt:

Satz 5.5 ([Papadimitriou, 1985]). *Die Klasse der Spiele gegen die Natur[3] entspricht der Klasse PSPACE.*

Insbesondere heißt dies, dass sich Spiele gegen die Natur mit polynomiellen Platz lösen lassen. In seinem Artikel gibt er auch sechs Beispiele für Spiele gegen die Natur an, die PSPACE-vollständig sind. Eines davon ist:

Definition 5.6 (Modified Stochastic SAT (SSAT')). *Gegeben ist eine Boolsche Formel B in konjunktiver Normalform über die Variablen $x_1, \ldots, x_n$ (n gerade) und eine rationale Konstante $q \in [0, 1]$.*

Die Frage ist, ob die Formel B mit Wahrscheinlichkeit größer gleich q erfüllt werden kann, wobei ungerade Variablen x_{2i-1} mit dem Existenzquantor belegt sind und gerade Variablen x_{2i} mit einem speziellen stochastischen Quantor $\mathcal{R}$, der folgendermaßen definiert ist.

Mit Wahrscheinlichkeit von jeweils $\frac{1}{4}$ gilt für $\mathcal{R}x$:

- *$\mathcal{R}x$ verhält sich wie der Existenzquantor $\exists x$.*
- *x wird auf* Wahr *fixiert.*
- *x wird auf* Falsch *fixiert.*
- *Die Formel ist unerfüllbar.*

Formal lautet die Frage also:

$$\exists x_1, \mathcal{R}x_2, \exists x_3, \ldots, \mathcal{R}x_n : Pr(B(x_1, \ldots, x_n) = \textit{Wahr}) \geq q$$

Satz 5.7 ([Papadimitriou, 1985]). *SSAT' ist PSPACE-vollständig.*

SSAT' lässt sich offensichtlich als Spiel gegen die Natur auffassen. Der Optimierer muss sich zunächst für Variable x_1 entscheiden, welchen Wahrheitswert er ihr zuweist. Daraufhin würfelt die Natur aus, wie sich der stochastische Quantor von Variable x_2 verhält. Nun entscheidet der Optimierer für die Variablen x_2 und x_3, welcher Wahrheitswert zugewiesen werden soll,[4]

[3] genauer die korrespondierenden Entscheidungsprobleme

[4] Je nachdem, wie sich der $\mathcal{R}$-Quantor von x_2 verhält, ist die Belegung für x_2 teilweise vorgegeben, bzw. es ist klar, dass die komplette Formel unerfüllbar ist.

usw. Anhand der Zuweisungen an die Variablen auf dem Pfad vom initialen Zustand (keine Variable zugewiesen) zu einem Endzustand (alle Variablen zugewiesen) kann dann bestimmt werden, ob B erfüllt ist. Falls ja, weisen wir dem Endzustand Kosten Eins zu, ansonsten Null. Offensichtlich ist B genau dann mit Wahrscheinlichkeit größer gleich q erfüllbar, wenn es eine Strategie mit Kosten mindestens q gibt.

Im nächsten Abschnitt definieren wir stochastische Flottenzuweisungsprobleme als Spiele gegen die Natur. Wir zeigen, dass bereits einfachste Varianten ebenfalls PSPACE-vollständig sind.

5.3 Komplexität

Wir definieren hier einige einfache stochastische Flottenzuweisungsprobleme und untersuchen ihre algorithmische Komplexität. Im Vergleich zu dem in Abschnitt 5.4 betrachteten Reparaturspiel treten hier keine Legausfälle auf und wir erlauben als Reparaturaktionen keine Legstreichungen und teilweise auch keine Startverschiebungen. Nichtsdestotrotz sind derartige stochastische Flottenzuweisungsprobleme zumeist bereits PSPACE-vollständig.

Definition 5.8 (SFAPfeas(f)). *Grundlage für eine Instanz von SFAPfeas(f) ist die deterministische Klasse FAP($Feas, Ac, f, Or$). Darüberhinaus sind eine Teilmenge $\mathcal{L}^S \subseteq \mathcal{L}$ der Legs, eine disjunkte Aufteilung der Planungsperiode in Intervalle $I_1, \ldots, I_n$ und eine rationale Konstante $q \in [0, 1]$ gegeben.*

SFAPfeas(f) ist ein Spiel gegen die Natur mit folgenden Regeln:

- *Optimierer und Natur führen abwechselnd jeweils n Züge $(O_1, N_1, \ldots, O_n, N_n)$ aus.*
- *In Zug O_i weist der Optimierer allen Legs, die im Intervall I_i starten, eine Flotte zu.*
- *In Zug N_i verlängert die Natur zufällig und unabhängig mit Wahrscheinlichkeit $\frac{1}{2}$ die Blockzeit aller Legs, die im Intervall I_i starten und zu $\mathcal{L}^S$ gehören, um Eins.*
- *Ein Endzustand ist zulässig, wenn die Flottenzuweisung, die sich aus dem Pfad vom initialen Zustand (kein Leg zugewiesen) zum Endzustand (alle Legs zugewiesen und teilweise gestört) definiert, eine zulässige (deterministische) Flottenzuweisung darstellt.*

Die Frage ist, ob es eine Strategie gibt, so dass sich mit Wahrscheinlichkeit mindestens q eine zulässige Flottenzuweisung ergibt.

Offensichtlich handelt es sich bei SFAPfeas(f) um ein Spiel gegen die Natur gemäß Definition 5.1.

Satz 5.9. *SFAPfeas(2) ist PSPACE-vollständig.*

Beweis. Als Spiel gegen die Natur gehört SFAPfeas(2) zu PSPACE.

Um die Vollständigkeit zu zeigen, reduzieren wir SSAT' aus Definition 5.6 auf SFAPfeas(2). Sei dazu

$$\exists x_1, \mathcal{R}x_2, \exists x_3, \ldots, \mathcal{R}x_n : Pr(B(x_1, \ldots, x_n) = \text{Wahr}) \geq q$$

eine Instanz von SSAT'. Wir verwenden hier fast die gleiche Transformation wie in Definition 3.26. Die Änderungen sind wie folgt:

- Das Paar, das Variable x_i repräsentiert, beginnt zum Zeitpunkt $4i$, d.h. wir fügen für Variable x_i das Paar $P_{x_i}(4i)$ dem Flugplan hinzu.

  Die Startzeit der beiden Eingangslegs q_j eines Paars, das Variable x_i repräsentiert, wird auf $4(i-1)$ gesetzt.

- Die Zeitpunkte aller übrigen Flughafenkonstrukte werden um $4(n+1)$ Zeiteinheiten nach hinten verschoben. Dadurch bleiben alle Blockzeiten im Flugplan echt positiv.

- $\mathcal{L}^S$ besteht aus den Eingangslegs q_j aller Paare, die eine $\mathcal{R}$-quantifizierten Variable x_{2i} repräsentieren.

- Die Planungsperiode wird in die Intervalle $I_0, \ldots, I_n$ unterteilt, wobei $I_i = [4i, 4(i+1)[$ für $i \in \{0, \ldots, n-1\}$ und $I_n = [4n, \infty[$ sind.

- q wird direkt von der SSAT'-Instanz übernommen.

Die Transformation lässt sich offensichtlich in polynomieller Zeit durchführen. Die Struktur des Flugplans wird gegenüber Definition 3.26 nicht geändert, nur die Variablen-Paare werden kaskadiert in der Zeit angeordnet. Abbildung 5.2 zeigt die Kaskadierung der ersten vier Variablen-Paare mit ihren Eingangslegs.

Nach Konstruktion muss der Optimierer in Intervall I_i entscheiden, wie er die Ausgangslegs vom Paar, das Variable x_i repräsentiert, zuweisen will. Unter den Voraussetzungen von Lemma 3.20 gilt dann:

- Falls x_i $\exists$-quantifiziert ist, gehören die Eingangslegs des Paars nicht zur Menge $\mathcal{L}^S$ und können deshalb keine verlängerte Blockzeit aufweisen. Der Optimierer kann hier also frei entscheiden, welche der beiden zulässigen Zuweisungen für die Ausgangslegs er wählt. Das Flughafenkonstrukt verhält sich wie ein Existenzquantor.

- Falls x_i $\mathcal{R}$-quantifiziert ist, gehören die Eingangslegs des Paars zur Menge $\mathcal{L}^S$ und können während des Intervalls I_{i-1} von der Natur jeweils um eine Zeiteinheit verlängert worden sein. Hier sind nun vier Fälle möglich, die jeweils mit Wahrscheinlichkeit $\frac{1}{4}$ eintreten:

q_1 gestört	q_2 gestört	zulässige Zuweisungen an (p_1, p_2)
Nein	Nein	(T,F), (F,T)
Nein	Ja	(F,T)
Ja	Nein	(T,F)
Ja	Ja	- (unzulässig)

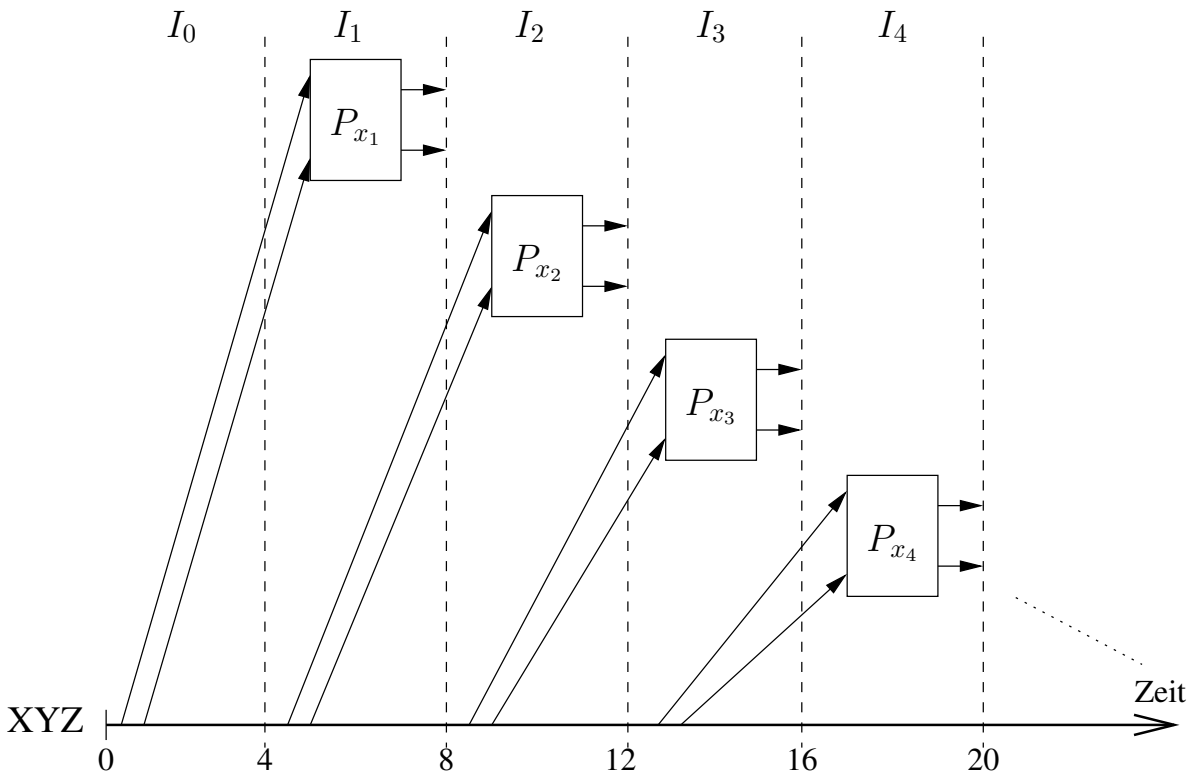


Abbildung 5.2: Kaskadierung der Variablen-Paare

Damit verhält sich das Flughafenkonstrukt exakt wie ein $\mathcal{R}$-Quantor. (Zur Erinnerung: Einer der Ausgänge p_j steht für das Literal x_i, der andere Ausgang für das Literal $\bar{x}_i$.)

Abbildung 5.3 zeigt die vier möglichen Fälle, wie sich die Eingangslegs eines Variablen-Paares einer $\mathcal{R}$-quantifizierten Variablen verspäten können, und welche Konsequenzen dies für die zulässigen Zuweisungen der Ausgangslegs hat.

Es verbleibt noch zu zeigen, dass die Boolsche Formel B genau dann mit Wahrscheinlichkeit mindestens q erfüllbar ist, wenn das oben konstruierte stochastische Flottenzuweisungsproblem mit Wahrscheinlichkeit mindestens q zulässig ist.

$\Rightarrow$ Nach dem oben Gesagten über die Variablen-Paare lässt sich wie im Beweis zu Satz 3.18 jede Strategie für die SSAT'-Instanz, die mit Wahrscheinlichkeit p erfüllbar ist, direkt in eine Strategie für das stochastische Flottenzuweisungsproblem transformieren, die mit derselben Wahrscheinlichkeit p zulässig ist.

Dazu muss man sich nur klar machen, dass eine Strategie S_B der SSAT'-Instanz *strukturell* mit einer Strategie S für das stochastische Flottenzuweisungsproblem übereinstimmt.

S_B ist ein Baum mit $n+1$ Ebenen, wobei sich in Ebene 1 nur die Wurzel befindet und in Ebene $n+1$ die Endzustände.

- Auf der Ebene $2i$, $1 \leq i \leq \frac{n}{2}$, befinden sich Naturknoten, die das Verhalten des $\mathcal{R}$-Quantors für Variable x_{2i} festlegen. Jeder dieser Naturknoten hat einen Ausgangsgrad von vier, wobei jede Kante mit Wahrscheinlichkeit $\frac{1}{4}$ gewählt wird.

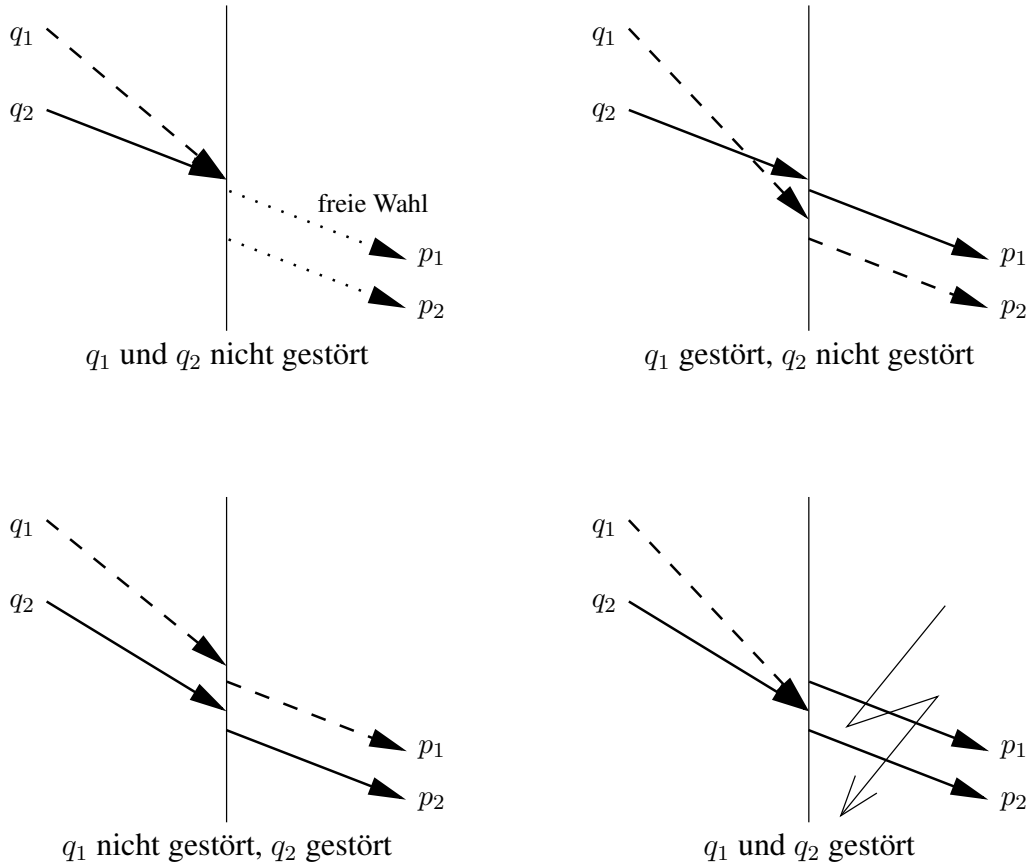


Abbildung 5.3: Die vier Störungsfälle für ein Variablen-Paar

- Auf Ebene $2i+1$, $0 \leq i < \frac{n}{2}$ befinden sich Optimierer-Knoten mit Ausgangsgrad eins, die die Belegung der Variablen x_{2i+1} und ggf. x_{2i} festlegen. Für x_{2i} muss dabei dann eine Belegung festgelegt werden, wenn der Vorgängernaturknoten auf Ebene $2i$ den $\mathcal{R}$-Quantor zu einem $\exists$-Quantor gemacht hat.

- Die Endzustände auf Ebene $n+1$ müssen ggf. noch die Belegung für Variable x_n festlegen und dann auswerten, ob die Boolsche Formel B mit der Variablenbelegung, die sich durch den Pfad von der Wurzel zum Endzustand definiert, erfüllend ist.

- S_B hat damit $4^{n/2} = 2^n$ viele Blätter.

Die Struktur von S, einer Strategie für das stochastische Flottenzuweisungsproblem, ist wie folgt. Da die Flottenzuweisungsinstanz die Planungsperiode in $n+1$ Intervalle aufteilt, müsste sich eigentlich ein Strategiebaum mit $2n+3$ Ebenen ergeben. Allerdings treten wegen der speziellen Konstruktion der Flottenzuweisungsinstanz Störungen nur in den ungeraden Intervallen $I_1, I_3, I_5, \ldots, I_{n-1}$ auf - nur dort starten Eingangslegs von $\mathcal{R}$-quantifizierten Variablen-Paaren. Somit können die Optimierer-Entscheidungen der Ebenen I_{2i} und I_{2i+1}, $0 \leq i < \frac{n}{2}$, zu einer Entscheidung zusammengefasst werden, und S ist ein Baum mit $n+1$ Ebenen, nummeriert von 1 bis $n+1$.

- In Ebene $2i$, $1 \leq i \leq \frac{n}{2}$, befinden sich wieder Naturknoten, die Legs aus $\mathcal{L}^S$ im Intervall I_{2i-1} stören. Dies sind genau die Eingangslegs des Variablen-Paars von Variable x_{2i}. Wie oben ausgeführt, können somit vier Störungsszenarien, jeweils mit Wahrscheinlichkeit $\frac{1}{4}$, auftreten. Die Knoten in Level $2i$ von Strategie S entsprechen also genau den Knoten in Level $2i$ von Strategie S_B.

- Auf Ebene $2i+1$, $0 \leq i < \frac{n}{2}$, befinden sich Optimierer-Knoten mit Ausgangsgrad eins, die die Zuweisung für die Legs, die in den Intervallen I_{2i} und I_{2i+1} starten, festlegen. Damit werden insbesondere die Ausgangslegs der Variablen-Paare für die Variablen x_{2i+1} und x_{2i} zugewiesen und damit analog zum Beweis von Satz 3.18 deren „Variablenbelegung“ festgelegt.

- In Ebene $n+1$ befinden sich die Endzustände. Diese müssen noch den Ausgangslegs des Variablen-Paars P_{x_n} und allen übrigen Legs der restlichen Flughafenkonstrukte eine Flotte zuweisen. Dies geschieht wie im Beweis zu Satz 3.18. Dann kann geprüft werden, ob es sich um eine zulässige Flottenzuweisung handelt oder nicht.

- Also hat auch die Strategie S insgesamt $4^{n/2} = 2^n$ viele Endzustände.

Die Strategien S und S_B sind demnach isomorph und eine Strategie S_B für die SSAT'-Instanz lässt sich direkt in eine Strategie S für die SFAPfeas-Instanz mit gleicher Zulässigkeitswahrscheinlichkeit überführen.

$\Leftarrow$ Sei S eine Strategie für das stochastische Flottenzuweisungsproblem mit Zulässigkeitswahrscheinlichkeit p. Sei E die Menge der *zulässigen* Endzustände von S. Jeder Pfad von der Wurzel der Strategie zu einem Zustand aus E beschreibt eine zulässige deterministische Flottenzuweisung. Für sie muss nach dem Beweis von Satz 3.18 gelten, dass jedes Variablen-Paar den Voraussetzungen von Lemma 3.20 genügt und dass die zu der Flottenzuweisung gehörende Variablenbelegung der Formel B, definiert durch die Ausgangsbelegung der Variablen-Paare, erfüllend ist.

Da also zumindest für die Teilstrategie S', die aus den Pfaden von der Wurzel von S zu den zulässigen Zuweisungen E besteht, die Voraussetzungen von Lemma 3.20 gelten, verhalten sich alle Variablen-Paare der Teilstrategie S' wie $\exists$- bzw. $\mathcal{R}$-Quantoren, und die Teilstrategie korrespondiert direkt zu einer Teilstrategie S'_B für die Boolsche Formel B, die nur erfüllende Variablenbelegungen enthält. Erweitert man S'_B beliebig zu einer vollständigen Strategie S_B, ist diese mindestens mit Wahrscheinlichkeit p erfüllend. □

Für das Problem SFAPfeas sind die Handlungsmöglichkeiten des Optimierers stark eingeschränkt: die einzige Entscheidung, die er treffen kann, ist die Zuweisung einer Flotte an ein Leg. Im Falle, dass, wie hier, Legs verspätet ankommen können, ist es natürlich, dem Optimierer als weitere Handlungsalternative zu erlauben, Legs verspätet starten zu lassen. Erlaubt man hier aber zu viele Freiheiten, wird zumindest das Zulässigkeitsproblem einfach:

Definition 5.10 (SFAPfeas$'(f)$). *Die Klasse SFAPfeas$'(f)$ ist fast wie die Klasse SFAPfeas(f) aus Definition 5.8 definiert. Die einzige Erweiterung besteht darin, dass der Optimierer im Zug O_i den Startzeitpunkt eines jeden Legs, das ursprünglich im Intervall I_i starten sollte,* beliebig weit *nach hinten verschieben darf.*

Satz 5.11. *SFAPfeas'*$(f) \in P$.

Beweis. Wir brauchen nur zu überprüfen, ob die Gesamtflugzeuganzahl F einer Instanz x von SFAPfeas'(f) ausreicht, um eine Instanz x' der *deterministischen* Variante FAPfeas(1) mit einer Flotte und F Flugzeugen zu lösen:

- Im hier betrachteten azyklischen Fall kann jeder Umlauf eines Flugzeugs von x' durch das Flugzeug einer beliebigen Flotte von x durchgeführt werden.
- Dabei spielen unterschiedliche Blockzeiten und eventuell auftretende Störungen keine Rolle, da die Startzeiten der Legs eines Umlaufs beliebig weit „auseinander" gezogen werden können, so dass die Anschlusslegs auf jeden Fall erreicht werden können.

Wir modellieren eine Instanz von FAPfeas(1) als gerichteten Multi-Graph $G = (V, E)$. Die Knotenmenge V entspricht den Flughäfen und für jedes Leg l fügen wir eine Kante (s_l^d, s_l^a) der Kantenmenge hinzu. Jeder kantendisjunkte Pfad in G entspricht damit einem legalen Umlauf eines Flugzeugs. Man beachte, dass Start- und Ankunftszeiten von Legs keine Rolle spielen, da wir die Startzeit eines jeden Legs beliebig weit nach hinten schieben dürfen, so dass ein Flugzeug, das auf einem Flughafen s landet, jedes von dort startende Leg als nächstes bedienen kann.

Die Frage nach der minimal benötigten Anzahl Flugzeuge ist damit offensichtlich identisch zu der Frage nach der minimalen Anzahl an kantendisjunkten Pfaden in G, die alle Kanten beinhalten. Diese Problem lässt sich in polynomieller Zeit mittels eines erweiterten Eulerpfad-Algorithmus lösen. Bezeichne dabei $out(v)$ den Ausgangsgrad eines Knoten v und $in(v)$ seinen Eingangsgrad:

1. Für jeden Knoten v mit $out(v) > in(v)$:
   - Konstruiere $out(v) - in(v)$ kantendisjunkte Pfade, die in v starten, nur bisher unbenutzte Kanten verwenden und nicht erweiterbar sind.

     Unter nicht erweiterbaren Pfaden verstehen wir hier Pfade, die in einem Knoten enden, der keine unbenutzten ausgehenden Kanten mehr besitzt.
2. Für jede noch unbenutzte Kante e:
   - Konstruiere einen nicht erweiterbaren kantendisjunkten Kreis K aus unbenutzten Kanten, der e enthält.
   - Verschmelze K dabei mit vorher konstruierten Kreisen, wenn diese einen gemeinsamen Knoten mit K besitzen.
   - Verfügt ein so konstruierter Kreis über einen gemeinsamen Knoten zu einem Pfad, so füge den Kreis in den Pfad ein.
3. Breche jeden noch existierenden Kreis K an einem beliebigen Knoten zu einem Pfad auf.

Die in Schritt 1. konstruierten Pfade sind offensichtlich nötig, um alle Kanten des Graphen abzudecken. Ist der Ausgangsgrad eines Knoten v größer als sein Eingangsgrad, können $out(v) - in(v)$ ausgehende Kanten keinen direkten Vorgänger besitzen und entsprechend viele Pfade müssen mindestens in dem Knoten v beginnen. Im Anschluss an 1. hat jeder Knoten in G eine balancierte Anzahl an eingehenden und ausgehenden unbenutzten Kanten. Die verbleibenden unbenutzten Kanten werden in Schritt 2. durch Kreise überdeckt. Das beschriebene Verschmelzen mit anderen Kreisen und Pfaden sorgt dafür, dass dabei nur eine minimale Anzahl an Kreisen entsteht. Die Kreise, die vor Schritt 3. übrig sind, stellen jeweils eine schwache Zusammenhangskomponente von G dar, die eine Eulertour besitzt. Offensichtlich ist für jede solche Zusammenhangskomponente ein weiterer Pfad notwendig, der in Schritt 3. erzeugt wird. Am Ende liefert der Algorithmus also die minimale Anzahl an kantendisjunkten Pfaden, die alle Kanten von G enthalten.

Der Algorithmus hat dabei offensichtlich polynomielle Laufzeit. □

Die Abflugzeit eines Legs beliebig lange nach hinten schieben zu können, ist nicht realistisch. Verlangt man für das Verschieben eines Legs Kosten von Eins pro Zeiteinheit, erhält man ein halbwegs praxisnahes Optimierungsproblem, das allerdings wieder PSPACE-vollständig ist.

Definition 5.12 (SFAP(f)). *Die Klasse SFAP(f) ist ein zur Klasse SFAPfeas$'(f)$ gehörendes Optimierungsproblem. Die Kosten eines Endzustand sind unendlich, wenn der Endzustand eine unzulässige Flottenzuweisung repräsentiert. Ansonsten bestimmen sich die Kosten eines Endzustands aus der Summe der Zeiteinheiten, um die der Optimierer die Startzeiten von Legs verschoben hat.*

Eine Strategie für eine Instanz von SFAP(f) hat damit nur dann endliche Kosten, wenn alle ihre Endzustände zulässige Flottenzuweisungen repräsentieren. Die Kosten der Strategie entsprechen der erwarteten Anzahl an Zeiteinheiten, um die die Abflugzeiten von Legs insgesamt nach hinten verschoben werden müssen.

Satz 5.13. *SFAP(2) ist PSPACE-vollständig.*

Beweis. SFAP(2) gehört als Spiel gegen die Natur zu PSPACE.

Um die Vollständigkeit zu zeigen, reduzieren wir QSAT auf SFAP(2). QSAT wird allgemein als das klassische PSPACE-vollständige Problem angesehen [Stockmeyer, 1976]. Sei dazu

$$\exists x_1, \forall x_2, \exists x_3, \ldots, \forall x_n : B(x_1, \ldots, x_n) = \text{Wahr}$$

eine QSAT-Instanz.

Die Transformation einer QSAT-Instanz in ein stochastisches Flottenzuweisungsproblem erfolgt wie im Beweis zu Satz 5.9. Die Rolle der $\mathcal{R}$-Quantoren übernehmen dabei natürlich die $\forall$-Quantoren. Außerdem werden sämtliche Landungen auf Flughafen XYZ auf einen neuen Flughafen XYZ' umgeleitet. Die Kostenschranke wird auf $q = \frac{n}{8}$ festgelegt.

Nach dem Beweis zu Satz 5.11 benötigt der so konstruierte Flugplan für eine zulässige Flottenzuweisung $2N$ Flugzeuge, da von Flughafen XYZ $2N$ Legs starten und keines landet.

N ist dabei die Anzahl an Paar-Konstrukten, die in der Flottenzuweisungsinstanz enthalten sind. Alle anderen Flughäfen bis auf XYZ' sind balanciert, auf XYZ' kommen nur Legs an, und der Flugplan besteht nur aus einer einzigen schwachen Zusammenhangskomponente. Alle verfügbaren $2N$ Flugzeuge der Flotten T und F müssen daher von Flughafen XYZ starten und jedes dieser Flugzeuge bedient genau eines der abfliegenden Legs.

Es ist damit klar, dass die Voraussetzungen der Lemmata 3.20 bis 3.24 für zulässige Zuweisungen gelten müssen und dass ein Flughafenkonstrukt genau dann keine Kosten verursacht, wenn es gemäß den Aussagen der Lemmata eingesetzt wird. Nur dann kann eine zulässige Zuweisung definiert werden, bei der der Abflug von Legs nicht verschoben werden muss.

Im Falle von Verspätungen, die nur bei $\forall$-quantifizierten Variablen-Paaren $P_{x_{2i}}(8i)$ auftreten können, gilt, dass unter den Voraussetzungen von Lemma 3.20 das Paar, wenn höchstens eines der Eingangslegs q_j verspätet ist, eine Zuweisung für die Ausgangslegs erzeugen kann, die keine Kosten verursacht (siehe Beweis zu Satz 5.9). Wenn beide Eingangslegs verspätet ankommen, muss das Ausgangsleg p_1 um eine Zeiteinheit nach hinten verschoben werden, damit eine zulässige Flottenzuweisung zustande kommen kann.[5] Dies folgt aus den Beobachtungen im Beweis von Satz 5.9 zu $\mathcal{R}$-quantifizierte Variablen und deren Variablen-Paaren. Somit tragen die $\frac{n}{2}$ Variablen-Paare mindestens erwartete Kosten von jeweils $\frac{1}{4}$ zu den Gesamtkosten bei. Mithin kann es (unter den Voraussetzungen von Lemma 3.20) keine Strategie mit Gesamtkosten echt kleiner q geben.

Es gilt nun, dass die QSAT-Instanz genau dann erfüllbar ist, wenn die konstruierte Flottenzuweisungsinstanz erwartete Kosten von (höchstens) q besitzt.

$\Rightarrow$ Sei S eine erfüllende Strategie der QSAT-Instanz. Diese definiert wie im Beweis zu Satz 5.9 sofort eine Teilstrategie S' für das Flottenzuweisungsproblem, wobei für jede $\forall$-quantifizierte Variable eines der Eingangslegs des korrespondierenden Variablen-Paares verspätet eintrifft. (Dies ist nach dem Beweis zu Satz 5.9 gleichbedeutend mit dem Fixieren der Variable auf T oder F.) Alle Endzustände von S' sind ferner zulässige Flottenzuweisungen und kommen ohne Startverschiebungen aus, generieren also keine Kosten.

Wir müssen S' nun noch für die Fälle, das keine oder zwei Verspätungen in einem Variablen-Paar auftreten, erweitern, um eine Gesamtstrategie für das Flottenzuweisungsproblem zu bekommen. Für den Fall, dass keine Verspätung im Paar $P_{x_{2i}}(8i)$ auftritt, kopieren wir einfach einen der beiden Äste für die Variable x_{2i}. Ohne Verspätung kann jede Ausgangsbelegung des Paares ohne Kosten realisiert werden. Für den Fall, dass beide Eingangslegs q_j verspätet sind, verschieben wir den Start von p_1 um eine Zeiteinheit und kopieren wieder einen der beiden Äste für die Variable x_{2i} in die Strategie S'. Durch das Verschieben von p_1 sind wieder beide Ausgangsbelegungen zulässig, und wir erhalten eine komplette Strategie S' für das Flottenzuweisungsproblem, das nur aus zulässigen Endknoten besteht. Die erwarteten Kosten belaufen sich auf genau q, da nur die $\frac{n}{2}$ Variablen-Paare von $\forall$-quantifizierten Variablen erwartete Kosten von jeweils $\frac{1}{4}$ generieren.

[5] Damit sich die hier auftretende Verzögerung von p_1 nicht auf den Rest des Flugplans auswirken kann, ersetzen wir das Leg p_1 durch zwei Legs p_1^d und p_1^a. Der Startflughafen von p_1^d entspricht dem Startflughafen von p_1 und der Zielflughafen von p_1^a dem Zielflughafen von p_1. Die beiden Legs begegnen sich auf einem neuen Flughafen und der Abstand zwischen der Ankunft von p_1^d und dem Abflug von p_1^a ist 3. Damit können Verzögerungen von 1 (und unterschiedliche Blockzeiten) kostenlos aufgefangen werden.

$\Leftarrow$ Sei S eine Strategie für das Flottenzuweisungsproblem mit Kosten höchstens q. Da die Kosten endlich sind, beschreibt jeder Endzustand eine zulässige Flottenzuweisung. Unter den Voraussetzungen von Lemma 3.20 muss S nach obigen Ausführungen genau Kosten q besitzen und die Kosten werden ausschließlich für Variablen-Paare mit zwei Eingangsverspätungen produziert. In der Teilstrategie S', die für jedes $\forall$-quantifizierte Variablen-Paar nur die Fälle mit genau einer Verspätung enthält, treten somit keine Legverschiebungen auf. Diese Teilstrategie korrespondiert dann aber direkt zu einer erfüllenden Strategie für die QSAT-Formel.

Bleibt noch die Frage zu klären, ob S vielleicht nicht die Voraussetzungen von Lemma 3.20 erfüllen muss. Dann ist es möglich, dass zu einem Paar P_1 zwei Flugzeuge der Flotte T fliegen. Selbst im Fall, dass sich beide Eingangslegs verspäten, muss keines der Ausgangslegs verschoben werden, um für P_1 eine zulässige Flottenzuweisung zu finden. Die erwarteten Kosten des Paares P_1 können also Null sein. Allerdings muss es zu einem Variablen-Paar P_1 mit zwei ankommenden Flugzeugen von Flotte T ein Paar P_2 mit zwei ankommenden Flugzeugen von Flotte F geben, da die Gesamtflugzeuganzahl ansonsten nicht ausreicht, um alle startenden Legs von Flughafen XYZ zu bedienen. Für dieses Paar muss dann aber das Leg p_2 *immer* um mindestens eine Zeiteinheit verschoben werden, das Paar P_2 hat also erwartete Kosten von mindestens Eins. Im Durchschnitt über beide Paare ergeben sich erwartete Kosten von mindestens $\frac{1}{2}$ pro Paar, was zu viel ist um unter der Kostenschranke q zu bleiben. Mit der gleichen Argumentation erzeugt auch eine Zuweisung von T an q_1 und F an q_2 mindestens erwartete Kosten von Eins. Der Fall, dass die Voraussetzungen von Lemma 3.20 nicht erfüllt sind, kann also für eine Strategie mit Kosten q nicht eintreten. □

5.4 Reparaturspiel

Beim Reparaturspiel handelt es sich um ein Spiel gegen die Natur mit speziellen Eigenschaften, wie sie für stochastische Flottenzuweisungsprobleme im Störungsmanagement aus Abschnitt 5.1.1 typisch sind.

5.4.1 Problemdefinition

Im Vergleich zu allgemeinen Spielen gegen die Natur gehen wir beim Reparaturspiel von folgenden Besonderheiten aus:

- Wir wollen die Planung für einen vorgegebenen Planungszeitraum durchführen. Dabei wollen wir zu *jedem* Zeitpunkt einen Plan für den kompletten Planungszeitraum besitzen, der im Fall, dass der „Normalfall“ eintritt, zulässig ist.
- Im Laufe des Planungszeitraums werden wir zu definierten Zeitpunkten über zufällige Störungen informiert, die in Form von diskreten Wahrscheinlichkeitsverteilungen gegeben sind. Als Reaktion darauf müssen wir bis zum Eintreten der nächsten Störungen den aktuellen Plan so reparieren, dass bis zum Ende der Planungsperiode wieder ein zulässiger Plan entsteht. Insbesondere reicht es an dieser Stelle nicht aus, nur die unmittelbar folgenden Aktionen zu planen.

- Wir bewerten Reparaturen anhand ihrer durchgeführten Änderungen bezüglich des aktuellen Plans. Nach einer Reparatur wird der reparierte Plan zum neuen aktuellen Plan, und später folgende Reparaturen werden bezüglich des neuen Plans bewertet. Das hat zur Folge, dass die Gesamtkosten von Reparaturen über den gesamten Planungszeitraum additiv sind und einmal verursachte Reparaturkosten nicht wieder ausgeglichen werden können.

Auf das stochastische Flottenzuweisungsproblem übertragen ergibt sich:

Definition 5.14 (Reparaturspiel für das stochastische Flottenzuweisungsproblem (REPGAME)). *Grundlage für eine Instanz von REPGAME ist ein azyklisches deterministisches Flottenzuweisungsproblem mit einer nicht notwendigerweise zulässigen Flottenzuweisung. Für jedes Leg ist eine diskrete Wahrscheinlichkeitsverteilung für mögliche Verspätungen und eine Wahrscheinlichkeit für den Ausfall des Legs gegeben, und die Planungsperiode ist in disjunkte Intervalle $I_1, \ldots, I_n$ aufgeteilt.*

REPGAME ist ein Spiel gegen die Natur mit folgenden Regeln:

- *Optimierer und Natur führen abwechselnd jeweils n Züge $(O_1, N_1, \ldots, O_n, N_n)$ aus.*
- *In Zug O_i generiert der Optimierer für alle Legs, die in den Intervallen I_i bis I_n starten, eine (deterministisch) zulässige Flottenzuweisung. Er darf dabei*
  - *Den Start von Legs nach hinten verschieben.*
  - *Die Flottenzuweisung von Legs ändern.*
  - *Legs komplett streichen.*

  Dabei verursacht er Kosten, die abhängig von den Änderungen bezüglich des aktuellen Plans sind.
- *In Zug N_i stört die Natur zufällig anhand der vorgegebenen Wahrscheinlichkeiten die Legs, die während des Intervalls I_i starten.*
- *Die Kosten von Endzuständen ergeben sich aus den aufaddierten Kosten aller vom Optimierer durchgeführten Umplanungen auf dem Weg von der Wurzel des Spielbaums hin zum jeweiligen Endzustand.*

Ziel des Spiels ist es, eine Strategie mit minimalen erwarteten Kosten zu finden. Da Strategien im Allgemeinen exponentiell groß sind, reicht es alternativ aus, neben den minimalen erwarteten Kosten nur den Plan des Optimierers in Zug O_1 anzugeben, da hierdurch die unmittelbar anstehenden Entscheidungen festgelegt werden.

Das so definierte Reparaturspiel ist eine Verallgemeinerung von Problem SFAP(f) aus Definition 5.12 und somit ebenfalls PSPACE-vollständig.

Algorithmus 3 mimav(Knoten v, Wert α, Wert β)

1: generiere alle Nachfolger $v_1, \ldots, v_b$ von Knoten v
2: **if** $b = 0$ **then**
3: **return** $c(v)$ {Blatt}
4: $cost = 0$
5: **for** $i = 1 \ldots b$ **do**
6: **if** v ist MAX-Knoten **then**
7: $\alpha = \max\{\alpha, \text{mimav}(v_i, \alpha, \beta)\}$
8: **if** $\alpha \geq \beta$ oder $i = b$ **then**
9: **return** α
10: **if** v ist MIN-Knoten **then**
11: $\beta = \min\{\beta, \text{mimav}(v_i, \alpha, \beta)\}$
12: **if** $\alpha \geq \beta$ oder $i = b$ **then**
13: **return** β
14: **if** v ist Natur-Knoten **then**
15: {Seien $w_1, \ldots, w_b$ die Wahrscheinlichkeiten der Knoten $v_1, \ldots, v_b$}
16: $\alpha' = \max\{\frac{\alpha - (cost + U \cdot \sum_{j=i+1}^{b} w_j)}{w_i}, L\}$
17: $\beta' = \min\{\frac{\beta - (cost + L \cdot \sum_{j=i+1}^{b} w_j)}{w_i}, U\}$
18: $cost = cost + w_i \cdot \text{mimav}(v_i, \alpha', \beta')$
19: **if** $cost + L \cdot \sum_{j=i+1}^{b} w_j \geq \beta$ **then**
20: **return** β
21: **if** $cost + U \cdot \sum_{j=i+1}^{b} w_j \leq \alpha$ **then**
22: **return** α
23: **if** $i = b$ **then**
24: **return** $cost$

5.4.2 Lösungsverfahren

5.4.2.1 Der mimav-Algorithmus

Für die Auswertung von Spielbäumen hat sich in der Praxis der Alphabeta-Algorithmus und seine Varianten bewährt. Er lässt sich verhältnismäßig leicht modifizieren, um auch Naturspieler berücksichtigen zu können. Besonders effizient lässt sich dabei der Naturspieler implementieren, wenn untere und obere Schranken auf die Kosten aller Knoten bekannt sind.

Seien nun U und L untere bzw. obere Schranken aller möglichen Kosten, die das Spiel annehmen kann. Wenn die Natur am Zug ist, besitzt jeder Zug der Natur eine vorgegebene Wahrscheinlichkeit, und die Kosten der Nachfolger werden gewichtet mit ihrer Wahrscheinlichkeit aufsummiert. Wenn wir uns also an einem Naturknoten befinden und die Untersuchung der ersten Nachfolger dazu geführt hat, dass auch im Extremfall der Wert des Knotens nicht mehr unter eine Schranke β fallen kann, haben wir einen Cut-off. Analoges gilt natürlich auch für die α-Schranke. Algorithmus 3 ist eine Beschreibung erweiterten Alphabeta-Verfahrens. Aufgerufen wird der Algorithmus mit mimav(Wurzel, L, U) und liefert als Ergebnis die optimalen erwarteten Kosten des Spielbaums.

Wir haben in Algorithmus 3 sowohl den Fall berücksichtigt, dass es sich bei dem Optimierer um einen Maximierer als auch einen Minimierer handeln kann. In der beschriebenen Form unterstützt der Algorithmus sogar den Fall, dass neben dem Optimierer und der Natur noch ein dritter Spieler, ein direkter Konkurrent des Optimierers, der diesem möglichst stark schaden will, am Spiel teilnimmt.

Die Auswertung der MAX- bzw. MIN-Knoten entspricht dem normalen vorgehen eines Alpha-beta-Algorithmus. Uns interessieren nur noch die genauen Kosten im Bereich zwischen α und β, und wenn ein Maximierer einen Nachfolger mit höheren Kosten als β findet, kann die Suche in diesem Knoten abgebrochen werden. Entsprechendes gilt für MIN-Knoten, wenn ein Nachfolger mit Kosten kleiner α gefunden wird.

Die Situation ist für Natur-Knoten ein wenig komplizierter. Die Kosten eines Natur-Knotens werden im Gegensatz zu MAX- bzw. MIN-Knoten nicht nur von *einem* Nachfolger definiert sondern ergeben sich aus einer gewichteten Summe der Kosten aller Nachfolger. Auch wenn einzelne Kosten unterhalb von α bzw. oberhalb von β liegen, heißt das nicht, dass dies auch für die gewichtete Summe gilt.

Schauen wir uns dazu die Situation nach der Bewertung des Nachfolgers v_i in Zeile 18 an. Die Knoten v_1 bis v_i sind bereits berechnet und die Summe ihrer gewichteten Kosten steht in $cost$. Die restlichen Knoten $v_{i+1}, \dots v_b$ werden nun noch mindestens $L \cdot \sum_{j=i+1}^{b} w_j$ zu dieser Summe beitragen. Liegt also diese untere Schranke auf die Naturknotenkosten oberhalb von β, können wir die Suche in Zeile 20 abbrechen. Entsprechend wird in Zeile 21 eine obere Schranke auf die Naturknotenkosten berechnet und die weitere Suche in Zeile 22 abgebrochen, wenn die obere Schranke unterhalb von α liegt.

Im Vergleich zu MAX- und MIN-Knoten müssen ferner die Nachfolger von Naturknoten mit veränderten Schranken α' und β' aufgerufen werden, da ja auch Nachfolgerkosten außerhalb des für den Naturknoten relevanten Intervalls zu Naturknotenkosten zwischen α und β führen können. Vor der Auswertung von Knoten v_i ist $UB = cost + U \cdot \sum_{j=i+1}^{b} w_j$ eine obere Schranke auf den Anteil, den die Nachfolgeknoten ohne v_i zur gewichteten Summe beitragen. Ist nun der Anteil von v_i zu dieser Summe kleiner als $\alpha - UB$, so wird die Naturknotensumme den Wert α garantiert nicht übersteigen können. Also interessieren uns für den Knoten v_i nur die genauen Kosten ab $\alpha' = \frac{\alpha - UB}{w_i}$, und diese Grenze wird in Zeile 16 bestimmt. Analog verläuft die Berechnung der oberen Grenze β' in Zeile 17. Das resultierende Intervall wird dabei noch zusätzlich an die überhaupt möglichen Kosten von Knoten angepasst, indem der Schnitt mit dem Intervall $[L, U]$ gebildet wird.

Mit Algorithmus 3 ist es prinzipiell möglich, jedes Spiel gegen die Natur auszuwerten. Allerdings muss der Algorithmus dabei mindestens die Knoten einer optimalen Strategie besuchen und im worst-case sogar den ganzen Spielbaum durchlaufen. In beiden Fällen erfordert dies aber exponentiellen Aufwand. Selbst verhältnismäßig kleine Spiele wie Schach lassen sich so nicht in akzeptabler Zeit exakt lösen.

In Abschnitt 5.4.2.3 präsentieren wir daher eine heuristische Variante des mimav-Algorithmus. Dafür benötigen wir allerdings ein Verfahren, um das deterministische Flottenzuweisungsproblem im Störungsmanagement zu lösen.

5.4.2.2 Modell für die deterministische Flottenumplanung

Die Definition des deterministischen Flottenumplanungsproblems entspricht der des Reparaturspiels ohne stochastische Einflüsse:

Definition 5.15 (Deterministisches Flottenumplanungsproblem (DETFAP)). *Grundlage für eine Instanz von DETFAP ist ein azyklisches deterministisches Flottenzuweisungsproblem mit einer nicht notwendigerweise zulässigen Flottenzuweisung.*

Es soll eine zulässige Flottenzuweisung berechnet werden, wobei

- *der Start von Legs nach hinten verschoben werden kann,*
- *die Flottenzuweisung von Legs geändert werden kann und*
- *Legs komplett gestrichen werden dürfen.*

Dabei fallen Kosten an, die abhängig von den Änderungen bezüglich des gegebenen Plans sind.

Ziel ist es, eine Flottenzuweisung mit minimalen Umplanungskosten zu finden.

Im Vergleich zu einem normalen Flottenzuweisungsproblem müssen wir also Legs verschieben und streichen können und eine geänderte Zielfunktion verwenden. Die Heuristiken aus Abschnitt 4.3 lassen sich nicht für das Flottenumplanungsproblem einsetzen, da keine zulässige initiale Flottenzuweisung bekannt ist. Alle IP-basierten Modelle lassen sich aber einfach erweitern, um die Anforderungen der Flottenumplanung zu unterstützen. Wir beschreiben hier nur die dafür notwendigen Modifikationen für das Time Space Network Modell aus Abschnitt 4.2. Die Ideen lassen sich leicht auf die übrigen Modelle übertragen.

Die Fähigkeit, Verschiebungen zu modellieren, geschieht auf folgende Weise: Wir schränken die Freiheitsgrade, wie ein Leg l verschoben werden kann, auf eine endliche (kleine) Menge $D_l = \{d_1, \ldots, d_{|D_l|}\} \subset \mathbb{N}_0$ von $|D_l|$ vielen Verschiebungen ein. Dabei ist d_1 immer gleich Null und steht für den Fall, dass das Leg nicht verschoben wird.

Wir kreieren nun eine neue Legmenge $\mathcal{L}'$, in die wir für jedes Leg $l \in \mathcal{L}$ der ursprünglichen Legmenge $|D_l|$ Kopien $l^D = \{l_1, \ldots, l_{|D_l|}\}$ von l aufnehmen. Alle Attribute (Startflughafen, Zielflughafen, zulässige Flotten, Blockzeiten, Mindestbodenzeiten) des ursprünglichen Legs l werden an die Kopien unverändert weitergereicht. Einzig die Startzeit von Leg l_i wird auf $t^d_{l_i,f} = t^d_{l,f} + d_i$ gesetzt, also um d_i Minuten nach hinten verschoben.

Um Legs streichen zu können, führen wir zu jedem Leg $l \in \mathcal{L}$ des ursprünglichen Flugplans eine zusätzliche binäre Variable $y_{l,*}$ ein, die anzeigt, ob ein Leg gestrichen werden soll ($y_{l,*} = 1$) oder nicht ($y_{l,*} = 0$).

Aus der neuen Legmenge $\mathcal{L}'$ wird nun ein normales Time Space Network wie in Abschnitt 4.2.2 aufgebaut. Mit den dort verwendeten Bezeichnungen lässt sich das deterministische Flottenumplanungsproblem wie folgt formulieren:

Modell 5.16 (DETFAP).

$$\textit{Minimiere} \sum_{l\in\mathcal{L}'}\sum_{f\in\mathcal{F}_l} c_{l,f}y_{l,f} + \sum_{l\in\mathcal{L}} c_C y_{l,*} \tag{5.1}$$

unter den Nebenbedingungen

$$y_{l,*} + \sum_{k\in l^D}\sum_{f\in\mathcal{F}_k} y_{k,f} = 1 \qquad \forall l \in \mathcal{L} \tag{5.2}$$

$$\sum_{(l,f)\in\mathcal{L}_v^a} y_{l,f} - \sum_{(l,f)\in\mathcal{L}_v^d} y_{l,f} + z_{v^-,v} - z_{v,v^+} = 0 \qquad \forall v \in V \tag{5.3}$$

$$\sum_{v\in V_f^\Delta} z_{*,v} \le N_f \qquad \forall f \in \mathcal{F} \tag{5.4}$$

$$y_{l,*} \in \{0,1\} \qquad \forall l \in \mathcal{L} \tag{5.5}$$

$$y_{l,f} \in \{0,1\} \qquad \forall l \in \mathcal{L}', f \in \mathcal{F}_l \tag{5.6}$$

$$z_{v,v^+} \in \mathbb{N}_0 \qquad \forall v \in V \tag{5.7}$$

$$z_{*,v} \in \mathbb{N}_0 \qquad \forall v \in \bigcup_{f\in\mathcal{F}} V_f^\Delta \tag{5.8}$$

Gegenüber dem normalen azyklischen Time Space Network Modell 4.5 haben sich hier nur die Zielfunktion und die Gleichungen (5.2) geändert. Die Gleichungen (5.2) stellen nun für jedes Leg $l \in \mathcal{L}$ der ursprünglichen Legmenge sicher, das höchstens einer Kopie von l aus der neuen Legmenge $\mathcal{L}'$ eine Flotte zugewiesen wird. Wird dabei die $y_{l,*}$-Variable auf Eins gesetzt, wird keine der zu Leg l gehörenden Flugkanten im Time Space Network verwendet und das Leg damit aus dem Flugplan gestrichen.

Für die veränderte Zielfunktion (5.1) führen wir zunächst die folgenden Bezeichnungen ein:

c_T	Kosten für die Verschiebung eines Legs um eine Zeiteinheit
c_E	Kosten für das Ändern der Flottenzuweisung eines Legs
c_C	Kosten für das Streichen eines Legs
$\delta_T(l)$	Anzahl Zeiteinheiten, um die das Leg $l \in \mathcal{L}'$ gegenüber seinem Original nach hinten verschoben worden ist
$\delta_E(l,f)$	$= 1$, falls für ein Leg $l \in \mathcal{L}'$ die Flottenzuweisung des Originals *nicht* Flugzeugtyp f ist; sonst ist $\delta_E(l,f) = 0$
$\delta_P(l,f)$	Änderungen auf der Erlösseite, wenn anstatt des Originals mit seiner ursprünglichen Flottenzuweisung das geänderte Leg $l \in \mathcal{L}'$ von Flotte f bedient wird

Die Umplanungskosten für ein Leg $l \in \mathcal{L}'$, wenn es von einem Flugzeug der Flotte f geflogen wird, belaufen sich damit zu

$$c_{l,f} = \delta_T(l)\cdot c_T + \delta_E(l,f)\cdot c_E + \delta_P(l,f)$$

und können in der Zielfunktion direkt den entsprechenden $y_{l,f}$-Variablen zugeordnet werden. Dazu kommen noch die Strafkosten für Legstreichungen, die von den $y_{l,*}$-Variablen angezeigt werden.

Unter der Annahme, dass keine weiteren Störungen auftreten, ist eine mit Modell 5.16 bestimmte Umplanung kostenminimal.

Aus der Definition der Zielfunktion ergibt sich in der Praxis ein sehr effizientes Laufzeitverhalten beim Lösen von deterministischen Flottenumplanungsproblemen. Da die Zielfunktion Abweichungen von einer existierenden, im Normalfall nur leicht gestörten Flottenzuweisung bestraft, zeigen unsere experimentellen Ergebnisse, dass sich auch große Instanzen in wenigen Sekunden lösen lassen, da die Zielfunktion implizit in Richtung einer fast zulässigen Lösung optimiert.

5.4.2.3 Heuristischer mimav-Algorithmus

Da eine vollständige Spielbaumauswertung nicht praktikabel ist, beschränkt man sich bei Alphabeta- und verwandten Algorithmen auf eine heuristische Baumauswertung, wobei die Suchtiefe und Suchbreite beschränkt werden. Daraus ergeben sich eine Reihe von zu beantwortenden Fragen:

- Wie sollen Suchtiefe und -breite beschränkt werden?

  Soll der Baum beispielsweise immer nur bis zu einer vorgegebenen Tiefe durchsucht werden, oder soll knotenabhängig die Suchtiefe angepasst werden?

- Im Falle, dass die Suchbreite beschränkt wird, welche Nachfolgeknoten sollen untersucht werden und welche ignoriert werden?

- Da man bei einer beschränkten Suchtiefe keine Endzustände im Spielbaum erreicht, muss geklärt werden, wie die Blätter des beschränkten Baums bewertet werden sollen.

Wir verwenden für unseren heuristischen mimav-Algorithmus eine feste Suchtiefe von derzeit 3. So halten sich die auftretenden Laufzeiten in erträglichen Grenzen.

Durch die besondere Struktur des Reparaturspiels ist eine Blattbewertung, auch wenn es sich nicht um einen Endzustand handelt, sehr einfach, da sich die Kosten eines Blattes aus der Summe der Umplanungskosten von der Wurzel bis zum Blatt zusammensetzen. Dabei wird implizit angenommen, dass nach dem vom Blatt repräsentierten Planungsintervall keine weiteren Störungen und damit Umplanungskosten mehr anfallen. Insofern handelt es sich bei dieser Blattbewertung natürlich um eine Heuristik.

Der interessanteste Punkt beim heuristischen mimav-Algorithmus ist die Art und Weise, wie die Suchbreite in jedem Knoten des Spielbaums beschränkt wird. Abhängig vom Knotentyp setzen wir dafür spezielle Zuggeneratoren ein.

Der Zuggenerator für die Natur untersucht, welche Abflüge im aktuellen Planungsintervall anstehen. Die atomaren Störungen „Streichung“ und „Verspätung um x Minuten„ werden für jedes Leg, das in diesem Zeitraum den Boden verlässt, generiert. Die atomaren Störungen werden gemäß ihren (vorgegebenen) Eintrittswahrscheinlichkeiten gewichtet. Im Prinzip würden wir jetzt gerne alle diese atomaren Störungen sowie alle ihre Kombinationen

untersuchen. Leider führt dies zu einer zu großen Anzahl möglicher Szenarien und damit zu einem zu hohen Knotengrad. Zurzeit beschränken wir daher die Suchbreite dadurch, dass wir nur atomare Störungen betrachten.

Wir gehen dabei wie folgt vor. Seien $l_1, \ldots, l_n$ die Legs, die im aktuellen Planungsintervall starten und damit von der Natur gestört werden können. Für Leg l_i seien S_i Störungen (Verspätungen oder Ausfälle) mit Eintrittswahrscheinlichkeiten $w_1^i, \ldots w_{S_i}^i$ gegeben. Mit $w_0^i = 1 - \sum_{s=1}^{S_i} w_s^i$ bezeichnen wir die Wahrscheinlichkeit, dass das Leg l_i nicht gestört wird. Wir gehen davon aus, dass die Störungen der Legs unabhängig voneinander sind, so dass sich in diesem Planungszeitraum insgesamt

$$\prod_{i=1}^{n}(S_i + 1)$$

verschiedene Szenarien ergeben. Bereits bei sechs Legs mit jeweils vier Störungen ergeben sich so $5^6 = 15625$ verschiedene Szenarien, die alle untersucht werden müssten. Dies ist bei weitem zu viel, so dass wir uns entschieden haben, beim Zuggenerator für die Natur ausschließlich die atomaren Szenarien zu betrachten, also die Szenarien, bei denen (höchstens) ein Leg gestört wird. Dann kommen wir mit

$$1 + \sum_{i=1}^{n} S_i$$

vielen Szenarien aus. Im Beispiel von oben wären das also nur noch $1 + 6 \cdot 4 = 25$ Fälle.

Jetzt verbleibt aber die Schwierigkeit, den eingeschränkten Szenarien brauchbare Eintrittswahrscheinlichkeiten zuzuweisen. Das Szenario, bei dem bei Leg l_i die s-te Störung auftritt und ansonsten alle übrigen Legs ungestört sind, tritt eigentlich nur mit Wahrscheinlichkeit $w_s^i \cdot \prod_{j \neq i} w_0^j$ auf. Würde man nur diese Wahrscheinlichkeiten verwenden, würde das Szenario, bei dem gar keine Störung auftritt, mit viel zu hoher Wahrscheinlichkeit eintreten.

Wir weisen daher dem Szenario S_s^i, bei dem nur für Leg l_i die s-te Störung auftritt, die folgende Wahrscheinlichkeit zu:

$$Pr(S_s^i) = \left(1 - \prod_{j=1}^{n} w_0^j\right) \frac{w_s^i}{\sum_{j=1}^{n} \sum_{t=1}^{S_j} w_t^j}$$

Damit bleibt für das Szenario S^0, bei dem keine Störung eintritt, die Wahrscheinlichkeit

$$\begin{aligned} Pr(S^0) &= 1 - \sum_{i=1}^{n} \sum_{s=1}^{S_i} Pr(S_s^i) \\ &= 1 - \left(\sum_{i=1}^{n} \sum_{s=1}^{S_i} \left(1 - \prod_{j=1}^{n} w_0^j \right) \frac{w_s^i}{\sum_{j=1}^{n} \sum_{t=1}^{S_j} w_t^j} \right) \\ &= 1 - \left(\left(1 - \prod_{j=1}^{n} w_0^j \right) \frac{\sum_{i=1}^{n} \sum_{s=1}^{S_i} w_s^i}{\sum_{j=1}^{n} \sum_{t=1}^{S_j} w_t^j} \right) \\ &= 1 - \left(1 - \prod_{j=1}^{n} w_0^j \right) \\ &= \prod_{j=1}^{n} w_0^j \end{aligned}$$

übrig, was genau der eigentlichen Eintrittswahrscheinlichkeit dieses Szenarios entspricht. Die Restwahrscheinlichkeit wird gemäß der w_s^i-Wahrscheinlichkeiten gleichmäßig auf die übrigen atomaren Szenarien S_s^i aufgeteilt.

Der Zuggenerator für den Optimierer erzeugt mehrere „gute“, aber unterschiedliche Reparaturalternativen für einen gestörten Plan. Sie führen alle so schnell und kostengünstig wie möglich zurück in den aktuellen gestörten Plan. Dabei kann jede Alternative aus einer Mehrzahl von Reparaturoperationen bestehen: aus Startverschiebungen, aus Neuzuweisungen von Flotten an Legs und aus Streichungen von Legs.

Die spezielle Struktur des Reparaturspiels gibt begründete Hoffnung, dass die so ausgewählten Reparaturalternativen zu guten Ergebnissen für den heuristischen mimav-Algorithmus führen. Da die Kosten einer zum jetzigen Zeitpunkt durchgeführten Umplanung später nicht mehr ausgeglichen werden können, führen sehr wahrscheinlich nur Umplanungen mit fast optimalen deterministischen Umplanungskosten auch zu geringen erwarteten Gesamtreparaturkosten.

Als Basis benutzen wir dabei das deterministische Flottenumplanungsmodell DETFAP 5.16 aus Abschnitt 5.4.2.2, um Reparaturalternativen zu erzeugen. Wir modifizieren jedoch die Branch&Bound Suche des IP-Lösers. Wenn eine neue ganzzahlige Lösung (also in unseren Worten eine gute und gültige Reparaturalternative) gefunden worden ist, wird diese Lösung gespeichert und durch Hinzufügen geeigneter Cuts für den Rest der Branch&Bound Suche für ungültig erklärt. Die Branch&Bound Suche terminiert, sobald c viele Reparaturalternativen gefunden worden sind. Zurzeit benutzen wir $c = 3$.

Heutige IP-Löser, wie die von uns eingesetzten Verfahren CPLEX und CBC, unterstützen das benutzergesteuerte Generieren von Cuts während der Branch&Bound Suche über ein Callback-Interface. Wann immer in einem Knoten während der Branch&Bound Suche eine ganzzahlige Lösung gefunden wird, werden die benutztereigenen Routinen aufgerufen, die die gefundene Lösung untersuchen können. Dabei können sie entscheiden, ob sie die ganzzahlige Lösung akzeptieren wollen oder nicht. Durch das Hinzufügen von benutzereigenen

Cuts (zusätzlichen Ungleichungen) zum Modell können sie so eine eigentlich aus IP-Sicht zulässige Lösung unzulässig machen und die Branch&Bound Suche zwingen, nach anderen ganzzahligen Lösungen zu suchen.

Der Vorteil dieser Vorgehensweise liegt darin, dass so zumindest für das Flottenumplanungsproblem das Finden von mehr als einer guten zulässigen Lösung kaum länger dauert als das Bestimmen nur einer Lösung mit dem Modell DETFAP. Beide Verfahren schaffen es durchschnittlich in unter 10 Sekunden einen gestörten Plan zu reparieren und eine oder eben auch mehrere gute Reparaturvorschläge zurückzuliefern.

Da wir nur eine verschwindend kleine Auswahl an Reparaturalternativen betrachten wollen, müssen diese zwei Anforderungen erfüllen, um für den heuristischen mimav-Algorithmus nützlich zu sein:

- Sie müssen die akuten Störungen kostengünstig reparieren, da nach den obigen Ausführungen einmal angefallene Reparaturkosten später nicht wettgemacht werden können.
- Die Reparaturalternativen sollten trotzdem strukturell möglichst verschieden sein, um in unterschiedliche Bereiche des Spielbaums vordringen zu können.

  Es ist sehr wahrscheinlich wenig hilfreich, wenn man zwei Reparaturalternativen geliefert bekommt, die sich nur darin unterscheiden, dass ein Leg 5 Zeiteinheiten verschoben worden ist. Zukünftige Störungen werden sich in beiden Reparaturalternativen sehr ähnlich auswirken.

Den ersten Punkt erreichen wir dadurch, dass wir zum Generieren der Reparaturalternativen das Modell DETFAP einsetzen, das kostenminimale Reparaturvorschläge erzeugt.

Die zweite Anforderung stellen wir dadurch sicher, dass wir sehr spezielle Cuts dem Modell hinzufügen, um eine gefundene Lösung und zu ihr strukturell ähnliche Lösungen von der weiteren Branch&Bound Suche auszuschließen.

Die Standardmethode, um speziell für IPs mit 0-1-Variablen eine eigentlich zulässige Lösung abzuschneiden, sieht wie folgt aus. Seien $x_1, \ldots, x_n$ die 0-1 Variablen eines IP-Modells, die in einer ganzzahligen Lösung den Wert 1 zugewiesen bekommen haben. Durch das Hinzufügen der Ungleichung

$$\sum_{i=1}^{n} x_i \leq n - 1 \tag{5.9}$$

wird die ganzzahlige Lösung unzulässig, da die linke Seite für die aktuelle ganzzahlige Lösung zu n ausgewertet wird.

Da die wesentlichen Entscheidungsvariablen $y_{l,f}$ im Modell DETFAP binär sind, ließe sich diese Technik prinzipiell zur Generierung von Cuts für DETFAP einsetzen. Allerdings würden so neben der aktuellen Lösung keine weiteren Lösungen für die Branch&Bound Suche abgeschnitten, insbesondere würde beispielsweise das Verschieben nur eines Legs sofort wieder zu einer zulässigen Lösung führen.

Um auch wirklich unterschiedliche Lösungsvorschläge zu bekommen, muss man also die zusätzlichen Cuts sorgfältig auswählen. Wir berücksichtigen für Cuts nur Entscheidungsvariablen von Legs, die anders gehandhabt werden (verschoben, neu zugewiesen oder gestrichen) als im Originalplan. Das heißt in unseren Cuts tauchen *grundsätzlich* nur die $y_{l,*}$-Variablen und $y_{l,f}$-Variablen, für die $\delta_T(l) > 0$ oder $\delta_E(l,f) = 1$ gilt, auf. Genau diese Variablen zeigen Reparaturaktionen am ursprünglichen Flugplan an, und da wir nach möglichst unterschiedlichen Reparaturalternativen suchen, sollten auch nur diese Variablen zur Entscheidung herangezogen werden.

Sei daher für eine aktuelle ganzzahlige Lösung während der Branch&Bound Suche im DETFAP-Modell Y die Menge an $y_{l,f}$ bzw. $y_{l,*}$-Variablen, die einen Wert von 1 haben und für eine Reparaturaktion stehen. Ist $Y = \emptyset$ heißt das, das der ursprüngliche Flugplan nicht repariert zu werden braucht, und wir generieren keine Cuts und geben uns mit der einen „Reparaturalternative“ zufrieden, da es aus der Sicht des Störungsmanagements keinen Sinn macht, einen zulässigen Plan auf Verdacht zu ändern.

Ansonsten bilden wir für die aktuelle Lösung Umlaufpläne für die Flugzeuge (siehe Bemerkung 3.13) und Partitionieren die Menge Y in Teilmengen $Y_*, Y_1, \ldots, Y_k$. In Y_* sind alle $y_{l,*}$-Variablen enthalten. In Y_i sind jeweils die $y_{l,f}$-Variablen zusammengefasst, die gemäß obiger Umlaufgenerierung von einem Flugzeug i bedient werden. Es gibt also k Flugzeuge, die geänderte Legs fliegen.

In der Menge Y_* befinden sich alle Variablen, die anzeigen, dass ein Leg gestrichen werden soll. Falls $Y_* \neq \emptyset$ ist, generieren wir für diese Menge einen Cut gemäß Ungleichung (5.9). Dadurch müssen in zukünftigen Lösungen andere Legs gestrichen werden.

Ferner konstruieren wir wie folgt einen Cut für jede der Mengen $Y_i = \{y_{l_1,f}, \ldots, y_{l_n,f}\}$:

$$\sum_{j=1}^{n} \sum_{m \in L(l_j)} y_{m,f} \leq n - 1,$$

wobei $L(l_j)$ für die Menge von Legs steht, die wie l_j Kopien eines gemeinsamen Originallegs sind, dass heißt sie stehen eigentlich für dasselbe Leg mit verschobenen Abflugzeiten. Bis auf die zusätzliche Summe über die Menge $L(l_j)$ handelt es sich also auch hierbei um einen Cut gemäß Ungleichung (5.9). Man beachte, dass in der Summe $\sum_{m \in L(l_j)} y_{m,f}$ wegen Gleichung (5.2) nur genau die Variable $y_{l_j,f}$ den Wert Eins besitzt und der Cut damit tatsächlich die aktuelle Lösung abschneidet.

Damit muss jedes der k Flugzeuge, das reparierte Legs fliegt, in zukünftigen Lösungen etwas anders machen. Dabei reicht es wegen der Summe über die Menge $L(l_j)$ nicht aus, einfach nur ein Leg etwas zu verschieben.

Durch die so generierten Cuts werden also neben der aktuellen Lösung auch strukturell ähnliche Lösungen von der weiteren Branch&Bound Suche ausgeschlossen, und wir erhalten gute, aber verschiedene Reparaturalternativen.

5.5 Experimentelle Ergebnisse

Ziel der experimentellen Untersuchung war es herauszufinden, ob die Berücksichtigung von stochastischen Daten zu einem verbesserten Störungsmanagement führt. Dabei sollte das etablierte deterministische Modell DETFAP aus Abschnitt 5.4.2.2 mit dem stochastischen Modell REPGAME aus Abschnitt 5.4.1 verglichen werden. REPGAME-Instanzen wurden dabei mit dem heuristischen mimav-Algorithmus aus Abschnitt 5.4.2.3 gelöst. Wir nennen im Folgenden DETFAP auch deterministisches Verfahren und REPGAME stochastisches Verfahren. Deterministisch/stochastisch bezieht sich dabei auf die Eingabedaten, beide Algorithmen arbeiten deterministisch.

Ein analytischer Vergleich der erzeugten Lösungen gestaltet sich schwierig, so dass wir uns mit unserem industriellen Partner Lufthansa Systems auf die Evaluation in einer Simulationsumgebung geeinigt haben.

5.5.1 Lösungsbewertung durch Simulation

Die Basis für unsere Simulation und die Eingabe für den Simulator stellt ein kontinentaler Lufthansa Flugplan dar. Zusätzlich stehen uns weitere Daten zur Verfügung, um alternative Pläne für die Reparatur berechnen und sich daraus ergebene Änderungen an Ticketerlösen bestimmen zu können. Der Plan besteht aus 20603 Flügen, die von 144 Flugzeugen aus 6 Flotten bedient werden.

Der Simulator teilt nun den Planungszeitraum in 15-minütige Intervalle auf und erzeugt nacheinander für jedes Intervall Störungen, die von einem der beiden Reparaturverfahren behoben werden müssen. Jeder Abflug des aktuellen Umlaufplans stellt ein mögliches Ereignis dar, wobei das betroffene Leg gestört werden kann, das heißt, es kann in unserem Fall um 30, 60 oder 120 Minuten verzögert werden oder auch ganz aus dem Flugplan gestrichen werden. Die Störungen treten dabei jeweils mit einer vorgegebenen Wahrscheinlichkeit auf. Bezeichnet T den aktuellen Simulationszeitpunkt, dann betrachtet und stört der Simulator alle Ereignisse im Intervall $[T, T + 15[$, übergibt die Störungen an ein Reparaturverfahren, wartet auf den reparierten Flugplan und geht 15 Minuten in der Zeit vorwärts.

Die Aufgabe der Reparaturverfahren besteht nun darin, die während der Simulation auftretenden Störungen möglichst kosteneffizient über den zu simulierenden Zeitraum hinweg zu beheben. Als Bewertung der Simulationsergebnisse haben wir vier relevante Maße während der Simulation protokolliert und über eine gewichtete Kostenfunktion zu unseren Reparaturkosten umgerechnet. Als Maße kamen dabei die Summe der benötigten Startverschiebungen in Minuten (TIM), die Anzahl der Wechsel eines Flugzeugtyps für ein Leg (ECH) und die Anzahl der Flug-Streichungen (CNL) zum Einsatz. Zusätzlich haben wir noch die Änderungen der Erlöse, die sich aus geänderten Passagierzahlen ergeben, berücksichtigt. Unser Industriepartner hat als Kostenfunktion c(TIM,ECH,CNL,Erlöse) = 50 TIM + 10000 ECH + 100000 CNL - Erlöse vorgeschlagen, mit der wir die Simulationsergebnisse der verschiedenen Reparaturverfahren verglichen haben.

Den Reparaturverfahren sind dabei die Parameter des Simulators bekannt, das heißt beide Verfahren setzen zur Bewertung ihrer Reparaturlösungen die selben Kostenparameter wie der

	deterministisches Reparaturverfahren				stochastisches Reparaturverfahren				
Datum	TIM	ECH	Erlöse	c	TIM	ECH	Erlöse	c	Δc
01/03	6270	0	0	313500	5850	0	0	292500	21000
01/04	6090	0	0	304500	6540	2	2219	344781	-40281
01/05	5820	0	0	291000	5910	0	0	295500	-4500
01/06	6240	2	-3717	335717	5910	2	-3717	319217	16500
01/07	5160	4	-28602	326602	5130	4	-28073	324573	2029
01/08	3600	4	-51674	271674	3570	4	-47784	266284	5390
01/09	5250	2	-28193	310693	5370	2	-6091	294591	16102
01/10	5730	0	0	286500	5940	0	0	297000	-10500
01/11	6270	4	-19915	373415	6090	4	-19915	364415	9000
01/12	6660	0	0	333000	6420	0	0	321000	12000
01/13	6750	4	-24790	402290	6810	2	-8782	369282	33008
01/14	5730	0	0	286500	5670	0	0	283500	3000
01/15	3390	0	0	169500	3270	0	0	163500	6000
01/16	5880	2	-4775	318775	5880	0	0	294000	24775
$\sum$	78840	22	-161666	4323666	78360	20	-112143	4230143	93523

Tabelle 5.1: Simulationsergebnisse für Wahrscheinlichkeiten 0.001/0.02/0.8/0.12

Simulator ein. Dem stochastischen Verfahren sind darüber hinaus auch die Intervalleinteilung des Simulators und die Wahrscheinlichkeiten der möglichen Störungen bekannt, so dass sich ein vollständig spezifiziertes Reparaturspiel REPGAME für die stochastische Flottenzuweisung ergibt.

5.5.2 Simulationsergebnisse

Wir haben den realen Flugplan der Lufthansa mit 144 Flugzeugen in 6 Flotten mit 20603 Legs in 14 aufeinander folgende Tage aufgeteilt und daraus 14 unabhängige Teilpläne als Testinstanzen generiert, für jeden der 14 Tage eine Instanz. Wie oben bereits erwähnt, wird dabei jede Instanz in 15-Minuten-Intervalle eingeteilt. Der Simulator kann verschiedene Reparaturverfahren zur Störungsbehebung einsetzen, und so haben wir das deterministische mit dem stochastischen Verfahren verglichen. Es ging dabei primär um die Frage, ob man für das gegebene Problem unter halbwegs realistischen Annahmen überhaupt Heuristiken finden kann, die eine deterministische Planung schlagen.

In einem ersten Testlauf haben wir die Wahrscheinlichkeiten für Legausfälle/Verspätungen um 120 Minuten/Verspätungen um 60 Minuten/Verspätungen um 30 Minuten auf 0.001/0.02/ 0.8/0.12 festgelegt. Die Ergebnisse finden sich in Tabelle 5.1. Die Spalten 2 bis 5 zeigen die Ergebnisse des deterministischen Verfahrens, die Spalten 6 bis 9 gehören zum stochastischen mimav-Reparaturverfahren. Flug-Streichungen (CNL) werden nicht gezeigt, da sie wegen der geringen Wahrscheinlichkeit während der Simulation nicht aufgetreten sind. Die Δc-Spalte zeigt die Kostendifferenz der c-Spalten der zwei Reparatur-Engines; positive Werte bedeuten, dass das stochastische Verfahren weniger Reparaturkosten verursacht hat als das deterministische. Wie aus der Tabelle ersichtlich wird, gewinnt das stochastische Verfahren 11 von 14 Tagen gegenüber dem deterministischen. Über alle 14 Tage zusammen kann der

# Proz.	Tag 01/03 Zeit	Tag 01/03 Speedup	Tag 01/04 Zeit	Tag 01/04 Speedup	Tag 01/05 Zeit	Tag 01/05 Speedup
1	226057	1.00	193933	1.00	219039	1.00
2	128608	1.76	111612	1.73	126915	1.73
4	68229	3.31	59987	3.23	66281	3.30
8	46675	4.84	40564	4.78	46065	4.75

Tabelle 5.2: Speedups der parallelisierten Baumsuche im mimav-Algorithmus

Datum	deterministisches Reparaturverfahren TIM	CNL	ECH	Erlöse	c	stochastisches Reparaturverfahren TIM	CNL	ECH	Erlöse	c	Δc
01/03	12210	8	2	-684	1431184	12120	8	2	-11374	1437374	-6190
01/04	11460	6	1	2028	1180972	11550	4	0	1145	976355	204617
01/05	10950	6	4	-24978	1212478	11340	8	5	-20407	1437407	-224929
01/06	13830	8	1	-5654	1507154	13470	8	1	-2567	1486067	21087
01/07	10530	7	2	-28385	1274885	10950	7	3	-33248	1310748	-35863
01/08	6000	4	4	-49501	789501	5430	2	4	-49784	561284	228217
01/09	11640	4	2	-29410	1031410	11570	4	2	-20977	1019477	6933
01/10	11730	6	3	-20403	1236903	11130	6	3	-25713	1212213	24690
01/11	12060	8	0	2003	1400997	12150	6	3	12119	1225381	175616
01/12	12030	6	2	-1971	1223471	11790	8	2	974	1408526	-185055
01/13	12630	8	0	580	1430920	12570	6	1	1036	1237464	193456
01/14	10410	6	4	-38488	1198988	10020	6	4	-36133	1177133	21855
01/15	5790	2	0	-1000	490500	5760	2	0	-1000	489000	1500
01/16	12270	5	0	-133	1113633	12090	4	0	257	1004243	109390
$\sum$	153540	84	25	-195996	16522996	152040	79	30	-185672	15987672	535324

Tabelle 5.3: Simulationsergebnisse für Wahrscheinlichkeiten 0.003/0.04/0.16/0.24

mimav-Algorithmus 2.21% an Reparaturkosten einsparen.

Was die Qualität der produzierten Reparaturpläne angeht, kann das stochastische Verfahren das DETFAP-Modell in allen drei Kategorien TIM, ECH und Erlöse schlagen. Das ist ein wenig überraschend, da wir erwartet haben, dass das stochastische Verfahren die „billigen“ TIMs einsetzen würde, um die teureren ECHs zu vermeiden und damit die Gesamtreparaturkosten zu senken.

Für jeden Reparaturschritt eines 15-Minuten-Intervalls benötigt der deterministische Algorithmus ungefähr 8 Sekunden. Der stochastische Verfahren ist ungefähr 200-mal langsamer, da es im Mittel 213 Suchknoten in jedem Schritt untersuchen muss. Eine Parallelisierung mittels dynamischer Lastverteilung brachte auch das stochastische Verfahren unter die wichtige Realzeitgrenze. In Tabelle 5.2 sind die Laufzeiten der Simulation in Sekunden und die erreichten Speedups mit bis zu 8 Prozessoren für die ersten drei Tage angegeben. Mit nur einem Prozessor beläuft sich die Simulationszeit mit dem mimav-Algorithmus für einen Tag simulierter Flugplan auf 2.5 Tage. Mit vier Prozessoren sinkt diese Zeit auf 18 Stunden.und damit unter die simulierte Planungsdauer.

In einem zweiten Lauf haben wir die Wahrscheinlichkeiten für Störungen auf 0.003/0.04/0.16/0.24 erhöht. Dann sieht ein typischer Testlauf wie in Tabelle 5.3 dargestellt aus.

Die Spalten 2 bis 6 von Tabelle 5.3 zeigen die Ergebnisse des deterministischen IP-Verfahrens aus Abschnitt 5.4.2.2, die Spalten 7 bis 11 gehören zu der stochastischen Heuristik aus Ab-

	Lauf 1	Lauf 2	Lauf 3	Lauf 4	Lauf 5
Woche 1	70293	27696	32261	-9238	-15799
Woche 2	8389	48778	11580	-1253	9144

Tabelle 5.4: Durchschnittliche eingesparte tägliche Reparaturkosten des stochastischen Verfahrens gegenüber dem deterministischen Verfahren

schnitt 5.4.2.3. Die Δc-Spalte zeigt die Kostendifferenz der c- Spalten der zwei Reparaturverfahren; positive Werte bedeuten, dass das stochastische Verfahren weniger Reparaturkosten verursacht hat als das deterministische. Wie aus der Tabelle ersichtlich wird, gewinnt das stochastische Verfahren 10 von 14 Tagen gegenüber dem deterministischen. Über alle 14 Tage zusammen kann die stochastische Heuristik 3.32% Reparaturkosten einsparen.

Hier können wir auch sehen, dass der mimav-Algorithmus weniger von den sehr teuren Flug-Streichungen produziert, dafür aber zusätzliche ECHs in Kauf nimmt. Eine interessante Beobachtung ist, dass man an den Ergebnissen erkennen kann, dass es sich beim 8. und 15. Januar um Sonntage handeln muss. Störungen haben an diesen Tagen signifikant geringere Auswirkungen, da der Schedule an Sonntagen nicht so „dicht gepackt“ ist.

Zwei bis drei Prozent Einsparungen sehen zwar schon schön aus, die gegebenen Daten klären aber noch nicht die Signifikanz der Ergebnisse. Obwohl an jedem Simulationstag fast 100 Entscheidungen getroffen werden, konnten wir die einzelnen Entscheidungen nicht direkt zur Klärung von Signifikanzfragen zu Hilfe nehmen, da die einzelnen Entscheidungen nicht unabhängig voneinander sind. Innerhalb eines Tages sind die Entscheidungen Folgeentscheidungen von Folgeentscheidungen, usw.

Die Resultate von einzelnen Tagen sind ebenfalls mit Vorsicht zu genießen. Zum einen scheinen sie nicht einer Normalverteilung zu genügen, zum anderen sind sie auch von der Struktur des Plans an einem bestimmten Tag abhängig. Wir nehmen daher Durchschnittswerte auf Wochenbasis als Messpunkte.

Wir haben deshalb zusätzliche Testläufe mit den bereits oben verwendeten Störungswahrscheinlichkeiten 0.003/0.04/0.16/0.24 durchgeführt. Dabei kam jeweils ein anderer Random-Seed zum Einsatz. Als Messpunkte wurden die durchschnittlichen Tagesreparaturkosten über eine komplette Woche genommen, wobei wir davon ausgehen, dass sich die beiden Wochen des Plans „hinreichend wenig“ in ihrer Struktur unterscheiden. Damit sind strukturelle Einflüsse (außer denjenigen, die auf die Pseudo-Zufälligkeit des Zufallsgenerators zurückzuführen sind) eliminiert, und wir erhalten insgesamt 10 Messpunkte, die in Tabelle 5.4 angegeben sind.

Ein Eintrag Woche i / Lauf j beschreibt den durchschnittlichen absoluten Tagesgewinn bzw. -verlust des stochastischen Verfahrens gegenüber dem deterministischen Verfahren in Woche i bei Simulationslauf j. Positive Werte sind günstig für das stochastische Verfahren. Der Mittelwert über diese Werte beträgt 18185 Einheiten, die Standardabweichung 26762. Gehen wir nun davon aus, dass die gegebenen Durchschnittswerte annähernd normalverteilt sind, ergibt sich mit Hilfe der t-Verteilung, dass mit 95% Sicherheit der stochastische Algorithmus besser ist als der deterministische.

5.6 Zusammenfassung

Wir haben in diesem Kapitel das stochastische Flottenzuweisungsproblem definiert und motiviert, dass es sich dabei um ein wichtiges Real-World-Problem handelt, mit dem Fluggesellschaften täglich im Störungsmanagement konfrontiert sind.

Wir konnten zeigen, dass selbst einfachste Untervarianten des stochastischen Flottenzuweisungsproblems PSPACE-vollständig sind.

Das stochastische Flottenzuweisungsproblem ist ein mehrstufiges Entscheidungsproblem unter Unsicherheit. Solche Probleme lassen sich als Spiel gegen die Natur modellieren und mit unserem generischen mimav-Algorithmus lösen. Die spezielle Struktur der stochastischen Flottenzuweisung, die wir Reparaturspiel nennen, erlaubt dabei, den mimav-Algorithmus erfolgreich heuristisch einzusetzen.

Unsere experimentellen Ergebnisse zeigen, dass durch das Spielen des Reparaturspiels robustere (Teil-)Pläne für die Flottenumplanung im Störungsmanagement erzeugt werden als mit etablierten deterministischen Verfahren. Unser vorausschauender Reparaturalgorithmus schlägt ein „kurzsichtiges“, exaktes IP-Verfahren statistisch signifikant in der vereinbarten Simulationsumgebung.

6

Integration von Ertragsmanagement und Flottenzuweisung

Dieses Kapitel beginnen wir mit einer Motivation zur Integration unterschiedlicher Planungsschritte in der Flugplanung. Wir gehen genauer auf die Aufgaben der Marktmodellierung und des Ertragsmanagements ein. Anschließend stellen wir drei Integrationsstrategien vor, die die Flottenzuweisung mit den beiden anderen Planungsphasen verbinden und auf diese Weise die Qualität des Planungsprozesses erhöhen.

6.1 Motivation

Durch die Integration von Planungsphasen kann die Gesamtqualität der Planung einer Fluggesellschaft weiter verbessert werden. Der Hauptgrund dafür liegt in den teils starken Abhängigkeiten, die zwischen den einzelnen Planungsphasen bestehen, und den vereinfachenden Annahmen, die zum Beispiel in der Flottenzuweisung über die Gewinnermittlung der Marktmodellierung angenommen werden.

Die meisten Modelle der Flottenzuweisung gehen von lokal berechenbaren Gewinnen aus, das heißt, es wird angenommen, dass sich der Gesamtgewinn aus unabhängigen Leggewinnen zusammensetzt, wobei die Leggewinne ausschließlich von der dem Leg zugewiesenen Flotte abhängig sind. In diesem Fall wäre die Gewinnfunktion durch die $p_{l,f}$-Werte exakt beschreibbar. Auf der Kostenseite kann diese Annahme als ausreichend realistisch angesehen werden, allerdings ist die Lage auf der Erlösseite im Allgemeinen komplizierter, wie das folgende Beispiel zeigt.

Auf einem Leg von FRA nach JFK würden gerne 300 Passagiere mitfliegen, die jeweils €500 für ein Ticket bezahlen. Beim Einsatz eines Flugzeugs mit 300 Sitzplätzen auf diesem

Leg ergeben sich also Erlöse von €150000. Beim Einsatz eines Flugzeuges mit nur 200 Passagieren können 100 Passagiere nicht mitgenommen werden und die Erlöse belaufen sich nur auf €100000. Was ist an dieser Betrachtung unrealistisch?

- Häufig benutzen Flugpassagiere nicht nur ein Leg, um ihrer Reise durchzuführen, sondern eine Folge von mehreren Legs, so genannte Reiserouten oder Itineraries.

  Ist nun aber ein Leg einer Reiseroute nicht verfügbar, wird die gesamte Reiseroute nicht gebucht und bezahlt. Somit kann das Nicht-Befördern der 100 Passagiere auch zu Erlösausfällen auf anderen Legs führen. Dieser Effekt wird *spill* genannt.

- Gleichzeitig wird durch das etwaige Freiwerden von Sitzplätzen auf anderen Legs zusätzlichen Passagieren die Möglichkeit gegeben, diese Legs zu buchen, was zusätzliche Erlöse bringen kann.

- Desweiteren werden sich die 100 Passagiere, die abgewiesen wurden, nach alternativen Reiserouten umsehen. Es ist zu erwarten, dass zumindest ein Teil von ihnen auf andere Legs der Fluggesellschaft ausweicht und dort für zusätzliche Erlöse sorgt. Dieser Effekt wird *recapture* genannt.

All diese Effekte, Spill&Recapture- oder Netzwerk-Effekte genannt, lassen sich *nicht* exakt durch rein lokale, lineare Gewinnbewertungen modellieren, und auch der Einsatz von verbindungsabhängigen Gewinnen reicht dafür nicht aus. Bei der Gewinnfunktion handelt es sich in der Realität um eine hochgradig nicht-lineare Funktion, die nur kompliziert zu berechnen ist. Es ist die Aufgabe der *Marktmodellierung*, zu einem Flugplan mit gegebener Flottenzuweisung den zu erwartenden Gewinn möglichst exakt zu bestimmen.

Je weiter man sich dem Tag der eigentlichen Umsetzung der Flugplanung nähert, desto genauer lassen sich aus den bereits eingegangenen Ticketbuchungen die vom Marktmodell nur recht grob geschätzten Passagierzahlen und erwarteten Erlöse vorhersagen. Die Planungsabteilung mit den genauesten Informationen in diesem Bereich ist das *Ertragsmanagement*. Beim Übergang zur kurzfristigen Planung sollte man daher in der Flottenzuweisungen die Daten des Ertragsmanagements zur Erlösermittlung verwenden.

6.2 Marktmodellierung

Die Aufgabe der Marktmodellierung ist es, in der lang- bis mittelfristigen Planung die zu erwartende Passagiernachfrage auf den eigenen Legs einer Fluggesellschaft möglichst genau vorherzusagen. Damit kann die Profitabilität eines Flugplans abgeschätzt werden. Die Marktbewertung ist damit die zentrale Kontrollinstanz für die im Laufe der Zeit entwickelten Flugplanszenarien (siehe auch Abbildung 1.2 auf Seite 3). Die parallel verlaufende Planung der Ressourcen greift immer wieder auf das Marktmodell zurück, um Veränderungen zu bewerten.

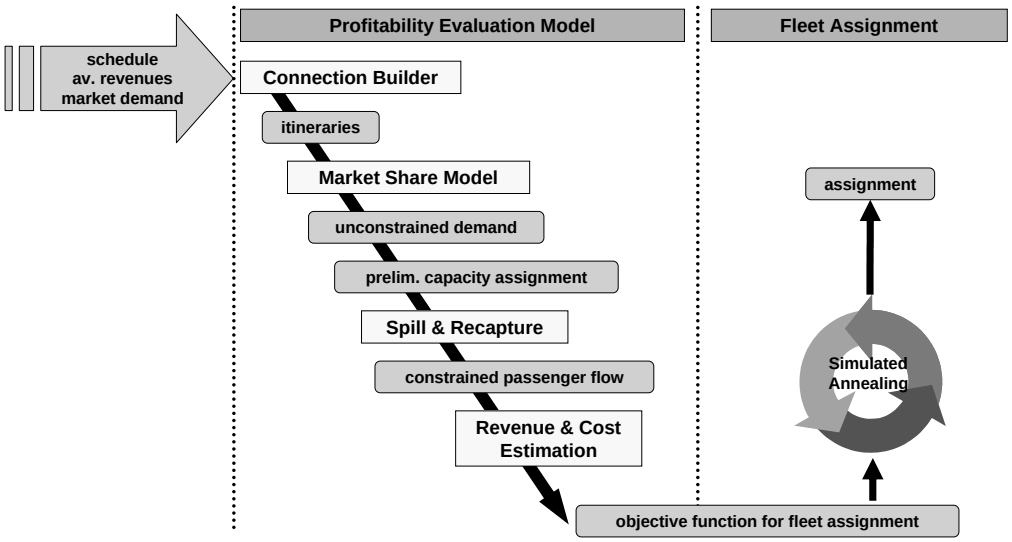


Abbildung 6.1: Die Vorgehensweise der Marktmodellierung

6.2.1 Überblick

Die Eingaben, Steuerungsparameter und Ausgaben der Marktmodellierung beschreibt die folgenden Tabelle.

	Marktmodellierung
Eingabe	Märkte (Städtepaare (O&D-Paare) mit Transportbedarf) Flugplan (Legs mit zugewiesener Flotte) ggf. Informationen zu konkurrierenden Fluggesellschaften und Verkehrsmitteln
Parameter	Regeln für den Aufbau der Reiserouten Parameter für die Verteilung der Marktnachfrage Definition der Kosten Definition der Erlöse
Ausgabe	Passagierfluss Gewinn = Erträge - Kosten des Flugplans

Ausgangspunkt ist der geschätzte (globale) Transportbedarf in Form von Märkten. Jeder Markt repräsentiert dabei die Menge an Personen, die von einer Stadt/Region (origin) zu einer anderen Stadt/Region (destination) gelangen wollen. Solch ein Markt wird daher auch O&D-Paar genannt. Auf der anderen Seite steht die verfügbare Transportkapazität in Form eines zugewiesenen Flugplans. Neben dem eigenen Flugnetzwerk bieten aber auch andere Fluggesellschaften Transportkapazitäten an, und man berücksichtigt teilweise sogar alternative Verkehrsmittel mit PKW und Eisenbahn. Die konkurrierenden Transportkapazitäten werden dabei meist vereinfacht berücksichtigt, um die Datengröße halbwegs erträglich zu halten.

In Abbildung 6.1 wird gezeigt, wie aus diesen Daten prinzipiell der auf das eigene Flugnetz entfallende Passagierfluss und die erwarteten Gewinne bestimmt werden. Zuerst werden für

jeden Markt die relevanten Reiserouten (Itineraries) bestimmt. Es gibt typischerweise sehr viele Möglichkeiten, um von A nach B zu gelangen, so dass hier eine Auswahl nach Attraktivität getroffen wird: die Ticketkosten sollten klein sein, die Reisezeit sollte kurz sein, es sollte nicht zu oft umgestiegen werden usw.

Der zentrale Punkt, auf den wir weiter unten noch näher eingehen, ist die anschließende Aufteilung der Marktnachfrage auf die generierten Reiserouten. Dabei kommen bei den meisten Fluggesellschaften Verhaltensmodelle zum Einsatz, die die Wahl eines einzelnen Passagiers von seinen persönlichen Präferenzen abhängig machen. In diesem Schritt bleiben Kapazitätsbeschränkungen der Legs zunächst unberücksichtigt und man erhält den unbeschränkten Transportbedarf (*unconstraint demand*) für jede Reiseroute.

Der unbeschränkte Transportbedarf übersteigt eventuell die Anzahl verfügbarer Sitzplätze auf einzelnen Legs. In diesem Fall müssen einige Passagiere einen alternative Reiseroute wählen. Ebenfalls über Verhaltensmodelle wird vorhergesagt, wie sich abgewiesene Passagiere verhalten und welche alternativen Reiserouten sie wählen werden (Spill&Recapture). Es ergibt sich ein erfüllbarer Passagierfluss im Flugnetz (constraint demand).

Damit ist für jedes Leg bekannt, wie viele Passagiere es verwenden werden. Daraus werden dann die zu erwartenden Erlöse berechnet und die variablen Kosten bestimmt. Die fixen Kosten eines Legs ergeben sich aus der dem Leg zugewiesenen Flotte und man kann den Gesamtgewinn des Flugplans bestimmen.

6.2.2 Verhaltensmodelle

Modelle des Passagierverhaltens (*passenger choice models*) basieren auf Präferenzen der Passagiere und erlauben Voraussagen, für welche der angebotenen Reisealternativen sich ein Passagier entscheiden wird. Im Marktmodell kommen dabei vor allem Discrete-Choice-Modelle, noch genauer so genannte Logit-Modelle, zum Einsatz.

Die nachfolgende Darstellung der Discrete-Choice-Theorie lehnt sich stark an das Buch von [Ben-Akiva and Bierlaire, 1985] an. Die Grundlagen werden auch im Buchkapitel in [Ben-Akiva and Bierlaire, 1999] vorgestellt. In [Koppelman and Sethi, 2000] werden diverse Logit-Modelle behandelt und deren Stärken und Schwächen miteinander verglichen.

Der Entscheidungsprozess eines Passagiers wird von vier Komponenten beschrieben: Entscheider, Alternativen, Attributen und Entscheidungsregel. Das Szenario ist wie folgt: Der Entscheider hat aus einer Menge der Alternativen eine auszuwählen. Die Entscheidung wird nach der Entscheidungsregel vorgenommen, die die Attribute der Alternativen auswertet. Jeder Entscheider besitzt eine Nutzenfunktion, die von den Attributen abhängt. Da dem Entscheider nicht unbedingt vollständige Informationen vorliegen, beeinflusst eine Zufallskomponente die ansonsten deterministische Nutzenfunktion.

Mit den folgenden Parametern

C_n	Auswahlmenge des Entscheiders n
i	Alternative $i \in C_n$ aus der Auswahlmenge von n
$x_{i,n,k}$	Attribut k für die Kombination (i, n)
β_k	Gewichtsparameter für Attribut k der Alternative i
$V_{i,n}$	deterministischer Anteil der Nutzenfunktion des Entscheiders n bei der Alternative i
$\varepsilon_{i,n}$	Zufallsterm
$U_{i,n}$	Nutzenfunktion des Entscheiders n bei der Alternative i

lässt sich die Entscheidungsfindung mittels der folgenden Funktionen beschreiben:

$$V_{i,n} = \sum_k \beta_k x_{i,n,k}$$
$$U_{i,n} = V_{i,n} + \varepsilon_{i,n}$$
$$Pr(i \mid C_n) = Pr(U_{i,n} = \max_{j \in C_n} U_{j,n})$$

Der deterministische Nutzen $V_{i,n}$ ist eine Summe der gewichteten Attribute einer Alternative. In der Nutzenfunktion $U_{i,n}$ des Entscheiders n kommt noch zusätzlich eine Komponente $\varepsilon_{i,n}$ vor, die die Zufälligkeit der Entscheidung abbilden soll. Der Passagier entscheidet sich damit für die wertvollste Alternative $U_{i,n} = \max_{j \in C_n} U_{j,n}$.

Wählt man $\varepsilon_{i,n}$ gemäß unabhängiger Gumbel-Verteilungen

$$F(\varepsilon) = e^{-e^{-\mu(\varepsilon - \eta)}}, \quad \mu > 0 \text{ (scale)}, \eta \text{ (location)}$$

ist die Wahrscheinlichkeit, dass ein Entscheider die Alternative i aus der Menge C_n auswählt, relativ einfach zu berechnen:

$$Pr(i \mid C_n) = \frac{e^{\mu V_{i,n}}}{\sum_{j \in C_n} e^{\mu V_{j,n}}}$$

Discrete-Choice-Modelle mit Gumbel-verteilten Zufallsvariablen bilden die Klasse der so genannten Logit-Modelle. Die Vorteile sind ihre einfache Implementierbarkeit und die Verfügbarkeit von Schätzalgorithmen, die es erlauben, alles Passagiere eines Marktes gleichzeitig auf die verfügbaren Reiserouten zu verteilen. Dem steht der Nachteil gegenüber, dass Logit-Modelle keine gegenseitigen Abhängigkeiten zwischen den Alternativen berücksichtigen können. Es gibt andere Discrete-Choice-Modelle, zum Beispiel Dogit- und Probit-Modelle, die mit Abhängigkeiten umgehen können und andere Zufallsverteilungen einsetzen. Diese haben sich in der Praxis allerdings wegen ihres deutlich höheren Rechenaufwands nicht durchsetzen können ([Müller-Bungart, 2003, Scheidler, 2003]).

Unser Industriepartner Lufthansa Systems, und damit auch wir, verwenden eine Logit-basierte Marktmodellierung wie in Abbildung 6.1. Die genaue Funktionsweise und die verwendeten Steuerungsparameter können in dieser Arbeit nicht beschrieben werden, da vor allem die Kalibrierung der Steuerparameter über die Brauchbarkeit der Marktmodellierung entscheidet und damit ein gut gehütetes Betriebsgeheimnis darstellt. In einem internen Dokument von Lufthansa Systems [Lefeld and Pölt, 1995] werden die fünf Bestandteile des Marktmodells detailliert beschrieben: Connection-Builder, das Logit-Modell, das Spill&Recapture Modul, das

Kosten-Modul und die Profitabilitätsbewertung. In [Sieber, 1995] wird die Spill&Recapture Komponente genauer vorgestellt. In der technischen Dokumentation [LufthansaSystems, 1997] werden Eingabedaten, Parameter und das Verhalten des Marktmodells beschrieben. Im Projektbericht [PARALOR, 1997] wird das Marktmodell und dessen Parallelisierung im Rahmen des BMBF-Projekts beschrieben.

Neben dem hier beschriebenen Logit-basierten Verfahren zur Marktmodellierung existieren noch eine Reihe anderer Verfahren mit unterschiedlichen Stärken und Schwächen. In seiner Dissertation gibt [Kniker, 1998] einen Überblick.

6.3 Ertragsmanagement

Der in diesem Abschnitt gegebene Überblick über die Verfahren im Ertragsmanagement (Revenue Management) folgt der Beschreibung im Buch von [Talluri and van Ryzin, 2005].

6.3.1 Übersicht

Die Aufgaben des Ertragsmanagements beinhalten die Aufteilung der Plätze auf einem Flug in unterschiedliche Kategorien, die Festlegung der Ticketpreise und die Steuerung der Ticketverkäufe. Diese Aufgaben werden während der ganzen Buchungsperiode wahrgenommen, das heißt zwischen der Veröffentlichung des Flugplans für eine Periode (Sommer-/Winterflugplan) und dem jeweiligen Flug. Das oberste Ziel des Ertragsmanagements ist dabei:

> Verkaufe das richtige Flugticket zum richtigen Zeitpunkt an die richtige Person zum richtigen Preis.

Daraus resultieren die wesentlichen Merkmale eines Ertragsmanagement-Systems:

Product differentiation Die Tickets werden nach Verkaufs- und Umbuchungsbedingungen (Business, Economy, Discount) mit bis zu zehn unterschiedlichen Preisklassen differenziert.

Dynamic pricing Während des Buchungszeitraums werden die Preise dynamisch an die Nachfrage angepasst.

Inventory control Die Verfügbarkeit der einzelnen Preisklassen wird ständig überwacht und unter der Berücksichtigung von Buchungsprognosen angepasst.

Im Bild 6.2 ist der Aufbau eines typischen Revenue Management Systems dargestellt. Zunächst werden die relevanten historische Daten werden gesammelt und für die Prognosen aufbereitet. Die Passagiernachfrage wird modelliert und daraus werden Buchungsprognosen erstellt, die laufend angepasst und verfeinert werden müssen.

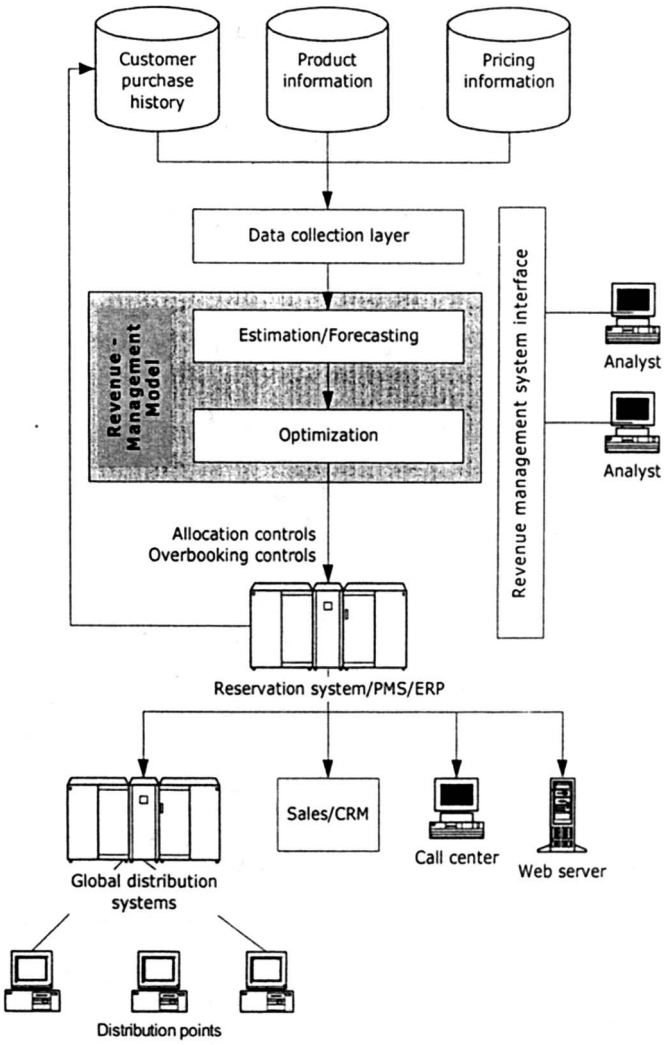


Abbildung 6.2: Ein Ertragsmanagementsystem (Quelle: [Talluri and van Ryzin, 2005])

Die Optimierung muss eine optimale Steuerung des Ticketverkaufs organisieren. Die aktuellen Preise werden festgelegt, und die Regeln für die Annahme oder das Abweisen von Buchungsanfragen werden aufgestellt.

Das Reservierungssystem empfängt über sämtliche Verkaufskanäle (Reisebüros, Call-Center, Internet) Anfragen und bedient sie nach der aktuell gültigen Steuerungsstrategie.

Die wichtigsten Bestandteile eines Ertragsmanagementsystems sind die Prognosemodelle für Buchungen und die Optimierungsverfahren zur Kapazitätssteuerung. Sie machen das eigentliche Revenue Management Modell aus.

Die Prognosemodelle müssen der Tatsache Rechnung tragen, dass unterschiedliche Kundengruppen ein unterschiedliches Buchungsverhalten an den Tag legen. In Abbildung 6.3 sind die zeitlichen Verläufe beispielhaft dargestellt. Business-Kunden buchen typischerweise erst ein Paar Tage vor Abflug und möchten vor allem flexible Umbuchungsmöglichkeiten erhalten. Economy- und Discount-Kunden buchen lange vor Abflug und nehmen diverse Restriktionen

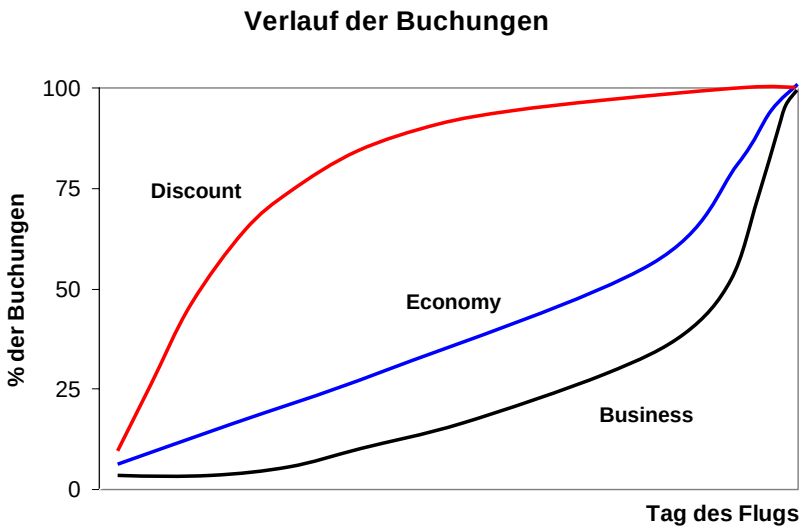


Abbildung 6.3: Buchungsverhalten unterschiedlicher Kundengruppen

(beispielsweise Wochenend-Regeln) in Kauf, vorausgesetzt der Ticketpreis bleibt niedrig.

6.3.2 Kapazitätssteuerung

Die grundsätzliche Frage des Revenue Managements lautet deswegen:

> Soll eine Economy- oder Discount-Buchungsanfrage jetzt akzeptiert werden oder lohnt es sich, den betreffenden Platz für eine spätere Business-Anfrage zu reservieren?

Verkauft man jetzt, besteht die Gefahr, dass im Flugzeug früh keinen freien Plätze mehr verfügbar sind und spätere gewinnbringende Business-Anfragen abgewiesen werden müssen. Verkauft man das Ticket jetzt nicht, kann es passieren, dass das Flugzeug halb-leer auf Reise gehen muss. In beiden Fällen verliert man Erlöse.

Im einfachsten Fall von nur zwei Klassen mit Preisen p_1 und p_2 ($p1 > p2$) und als Wahrscheinlichkeitsverteilungen gegebenen Bedarfen D_1 und D_2 kann der Platz x an die Anfrage zum Preis p_2 vergeben werden, falls gilt:

$$p_2 \geq p_1 \cdot Pr(D_1 \geq x)$$

Der Preis p_2 übersteigt also den erzielbaren erwarteten Preis für diesen Sitz in der Klasse 1. Da die Funktion $Pr(D_1 \geq x)$ fallend in x ist, existiert ein optimaler Wert x_1^* mit:

$$p_2 = p_1 \cdot Pr(D_1 \geq x_1^*)$$

Bei einer kontinuierlichen Wahrscheinlichkeitsfunktion $F(x)$ für D_1 kann dieser optimale Wert mit Littlewood's Regel bestimmt werden ([Littlewood, 1972]):

$$x_1^* = F^{-1}(1 - p2/p1)$$

Entsprechend dieser Regel kann eine Buchungsgrenze von x_1^* Plätzen definiert werden, die die Klasse-1 vor Klasse-2 Buchungen schützt: Es werden x_1^* Plätze exklusiv für Klasse-1-Passagiere reserviert. Bei sich ändernden Preisen oder Bedarfsverteilungen muss diese Grenze natürlich entsprechend angepasst werden.

Eine Verallgemeinerung dieser Regel auf n Klassen wurde von [Belobaba, 1989] vorgeschlagen. Die heuristische Strategie EMSR-b (expected marginal seat revenue - version b) findet eine sehr breite Verwendung in Ertragsmanagementsystemen vieler Fluggesellschaften.

Dabei geht man bei n Klassen mit Preisen $p_1 > p_2 > \ldots > p_n$ und als Zufallsvariablen gegebenen Klassenbedarfen $D_1, \ldots, D_n$ davon aus, dass die Ticketanfragen in umgekehrter Klassenreihenfolge $n, \ldots, 2, 1$ das Ertragsmanagementsystem erreichen. Nach den billigen Tickets wird also zuerst verlangt.

In der Runde mit Ticketanfragen von Klasse $j+1$ suchen wir nach einer Buchungsgrenze x_j für die Klassen $1, \ldots, j$. Der Gesamtbedarf dieser Klassen ist $S_j = \sum_{k=1}^{j} D_k$, ebenfalls eine Zufallsvariable. Der gewichtete Durchschnittspreis der Klassen $1, \ldots, j$ ist

$$\bar{p}_j = \frac{\sum_{k=1}^{j} p_k \cdot E[D_k]}{\sum_{k=1}^{j} E[D_k]}$$

Die Buchungsgrenze x_j wird daher auf

$$Pr(S_j > x_j) = \frac{p_{j+1}}{\bar{p}_j}$$

festgelegt. In jeder Runde kann so bestimmt werden, wie viele Buchungsanfragen in der jeweiligen Preisklasse akzeptiert werden können.

6.3.3 Bid prices

Auch das Ertragsmanagement steht vor dem Problem, dass die Sitzplätze eines Legs häufig nicht unabhängig von anderen Legs vergeben werden können. Die Problematik ist ähnlich wie bei der linearen Gewinnfunktion der Flottenzuweisung in Abschnitt 6.1.

Bei der Anfrage eines Kunden nach einer Reiseroute, für die er p bezahlen will, steht das Ertragsmanagementsystem vor dem Problem, den Ticketpreis auf die verschiedenen Legs der Reiseroute aufteilen zu müssen. Ist das geschafft, kann es aber passieren, dass das System dem Kunden nur Plätze für einige Legs der Reiseroute verkaufen möchte. Der Kunde hat aber ein Paketangebot gemacht und will entweder die komplette Reiseroute fliegen können oder auf alle Flüge verzichten.

In diesem Fall muss die Kapazitätskontrolle auf der Netzwerkebene ausgeführt werden und wir kommen in den Bereich des *O&D-Revenue Managements* (origin-destination revenue management). Von den existierenden Arten der Netzwerkkontrolle möchten wir in diesem Abschnitt auf die *bid-price* Steuerung eingehen.

Ein bid-price definiert eine Preisschwelle für jeden Sitz auf Legs im Netzwerk. Bei einer Buchungsanfrage für eine Reiseverbindung, die eventuell aus mehreren Legs besteht, wird deren Preis mit der Summe der bid-prices der einzelnen Legs verglichen und die Anfrage dann akzeptiert, wenn der gebotene Preis höher ist.

Bid-price-Kontrollstrategien sind nicht immer optimal, liefern aber eine gute Approximation der optimalen Kontrolle [Talluri and van Ryzin, 1998].

Bid-prices können mit unterschiedlichen Methoden berechnet werden:

- Globale Approximationsmethoden:
  - deterministisches lineares Modell
  - probabilistisches oder randomisiertes lineares Modell
- Dekompositionsmethoden:
  - *OD factors* Methode
  - *prorated EMSR*
  - *DAVN: displacement-adjusted virtual nesting*
  - Dynamische Programmierung
  - *iterative DAVN*
  - *iterative prorated EMSR*

Wir gehen hier nicht genauer auf die einzelnen Verfahren ein. Die EMSR-Varianten basieren auf den Beschreibungen des vorherigen Abschnitts, da neben den Buchungsgrenzen implizit auch die Wertigkeit von Sitzen ermittelt werden kann. In Abschnitt 6.4.2 verwenden wir das deterministische lineare Modell, um eine verbesserte Zielfunktion für das Flottenzuweisungsproblem zu erhalten.

Das Ergebnis dieser Methoden ist ein bid-price für jedes Legs, genauer jeden Sitz eines Legs, im Netzwerk. Die bid-prices geben damit sehr genau den Wert von Sitzplatzkapazitäten auf den einzelnen Legs wieder. Dies ist eine Information, die wir in Abschnitt 6.4.3 benutzen, um eine genauere Erlösschätzung, als sie das Marktmodell liefern kann, dem heuristischen Flottenzuweisungsoptimierer verfügbar zu machen.

6.4 Integrationsstrategien

Allen nun vorgestellten Integrationsstrategien ist gemein, dass sie dem Flottenzuweisungsproblem eine verbesserte Zielfunktion liefern sollen. Grundsätzlich werden dabei Zwischenlösungen während der Optimierung eines Flottenzuweisungsproblems an die integrierten Algorithmen übergeben, die darauf aufbauend eine neue, an die aktuelle Zwischenlösung angepasste Zielfunktion zurückliefern.

Ein Wechsel der Zielfunktion während eines Optimierungslaufs ist für IP-Löser praktisch kaum zu bewerkstelligen. Ferner stehen während der Branch&Bound-Suche nicht immer zulässige Zwischenlösungen zur Verfügung, so dass die hier beschriebenen Integrationsstrategien ausschließlich mit den heuristischen Lokale Suche Verfahren aus Abschnitt 4.3 realisierbar sind.

6.4.1 Marktmodellierung und Flottenzuweisung

Das Marktmodell wird bereits dazu verwendet, die (lineare) Zielfunktion eines Flottenzuweisungsproblems aufzustellen. Das Vorgehen ist in Abbildung 6.1 auf Seite 169 dargestellt. Eigentlich berechnet das Marktmodell für einen gegebenen Flugplan inklusive Flottenzuweisung nur den erwarteten Gewinn (pro Leg). Wir benötigen aber zusätzlich für jedes Legs den Gewinn, der sich aus den alternativ möglichen Flottenzuweisungen an dieses Leg ergibt. Hierzu wird die Flottenzuweisung jeweils eines Legs geändert und aus den sich ergebenden Kosten- und Erlösänderungen die Gewinndifferenz zur Originalzuweisung bestimmt. Dies liefert den Gewinn, wenn das betroffene Leg von einer alternativen Flotte bedient wird, und wir erhalten so alle $p_{l,f}$-Werte, die unsere Zielfunktion definieren.

Wie wir in Abschnitt 6.1 gesehen haben, verhält sich die so definierte Zielfunktion aber nur dann wie das Marktmodell, wenn sich gegenüber der zur Berechnung verwendeten Flottenzuweisung die Flotte höchstens eines Legs ändert. Bei mehr als einer Änderung können die Zielfunktion und das Marktmodell eine Zuweisung unterschiedlich bewerten, da nur das Marktmodell alle Netzwerkeffekte berücksichtigen kann.

Da die Lokale Suche Verfahren nicht auf eine lineare Zielfunktion angewiesen sind, ist eine erste Idee, die Lösungsbewertung in der Heuristik einfach komplett dem Marktmodell zu überlassen und so immer mit einer Marktmodell-konformen Lösungsbewertung zu arbeiten. Allerdings verhindern die vor allem im Vergleich zur Nachbarschaftgenerierung sehr langen Laufzeiten des Marktmodells einen solchen Ansatz. Es können pro Sekunde hunderte von Nachbarn generiert und mit einer linearen Zielfunktion bewertet werden, die Zeiten für eine Lösungsbewertung durch das Marktmodell liegen im Minutenbereich.

Daher verwendet diese Integrationsstrategie eine periodische Kommunikation zwischen der Lokalen Suche Heuristik und dem Marktmodell. Abbildung 6.4 zeigt die Vorgehensweise.

Nachdem zunächst in der ersten Phase die Marktmodellierung eine lineare Zielfunktion für das Flottenzuweisungsproblem aufgestellt hat, beginnt die Optimierung der Flottenzuweisung durch den Simulated Annealing Algorithmus. Die Kommunikation findet jeweils nach einer abgeschlossenen Temperaturstufe des Simulated Annealing Algorithmus statt. Die aktuelle

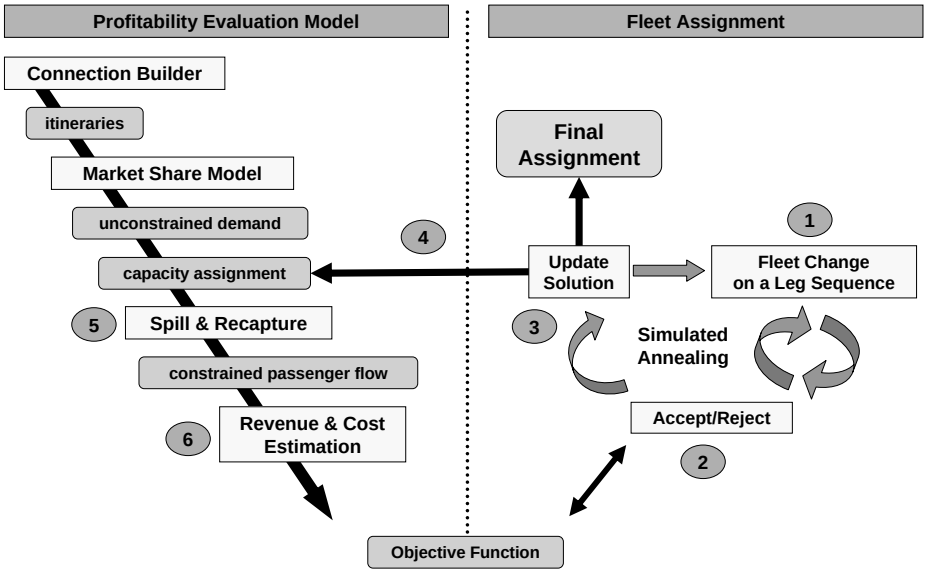


Abbildung 6.4: Kopplung der Marktmodellierung und der Flottenzuweisung

Zuweisung wird an das Marktmodell gesendet. Die empfangene Lösung wird im Marktmodell neu bewertet, indem die neuen Kapazitäten auf den Flügen bei der Berechnung des Passagierflusses berücksichtigt werden. Da die Lösungen aus der Flottenzuweisung keine Startzeiten verändern, sind eine erneute Berechnung der Reiserouten und die Bestimmung des unconstraint demand nicht notwendig. Nur durch die Spill&Recapture-Effekte ergibt sich eine veränderte Zielfunktion, die wie zu Beginn aufgestellt wird und anschließend dem Simulated Annealing Algorithmus übergeben wird.

Die Häufigkeit der Kommunikation und die Anzahl der Kommunikationsrunden ist veränderbar:

- *Beginn der Kommunikation:* Nach Erreichen einer vorgegebenen Akzeptanzrate im Simulated Annealing Algorithmus. Dadurch kann z.B. vorgegeben werden, dass in der Anfangsphase der Optimierung, in der die Lösung noch sehr stark verändert wird, keine Kommunikation stattfinden soll. Der Standardwert für die Akzeptanzrate liegt bei 20%.
- *Häufigkeit der Kommunikation:* Anzahl der Temperaturstufen zwischen zwei Kommunikationsphasen. Das Kommunikationsmuster kann auch dynamisch verändert werden. In der Anfangsphase kann zum Beispiel nach jeder 10. Temperaturstufe kommuniziert werden, später wird immer häufiger kommuniziert, bis am Ende der Optimierung nach jeder Stufe die Lösung verschickt wird.
- *Schwelle für die Kommunikation:* Die Kommunikation soll erst stattfinden, wenn eine vorgegebene Anzahl von Flügen eine andere Flotte zugewiesen bekommen hat. Dadurch wird erreicht, dass eine zeitintensive Kommunikation nicht für ganz wenige Veränderungen in der Flottenzuweisung angestoßen wird.
- *Anzahl der Kommunikationsrunden:* Die Anzahl kann vom Systembenutzer vorgegeben werden. Damit kann erreicht werden, dass für bestimmte Analysen eine schnelle

Antwort geliefert wird und nicht unbedingt die vollständige Konvergenz des Verfahrens abgewartet werden muss.

Als Hauptvorteil dieser Kopplung kann die bessere Berücksichtigung der Netzwerkeffekte genannt werden. Diese Effekte werden vom Marktmodell berücksichtigt und explizit modelliert. Durch die angepasste Zielfunktion kann die Lokale Suche die Auswirkung der Flottenänderung im Netzwerk erkennen und durch weitere Optimierung darauf reagieren. Die Ergebnisse der Experimente mit dem neuen System werden im Abschnitt 6.5 präsentiert.

Der Nachteil des Verfahrens liegt hauptsächlich in der zeitintensiven Neubewertung durch das Marktmodell. Typischerweise werden ca. 10-20% der Legs mit einer anderen Flotte geflogen und dementsprechend müssen Passagiere auf sämtlichen Reiserouten, die diese Flüge benutzen, neu bewertet werden. Die im nächsten Abschnitt vorgestellte Vorgehensweise versucht, einen Kompromiss zwischen der Genauigkeit der Voraussagen über den Passagierfluss und der Berechnungsgeschwindigkeit zu finden.

6.4.2 Berücksichtigung des Passagierflusses

Ziel dieser Integrationsstrategie ist es, die aufwendige Neuberechnung der Zielfunktion durch das Marktmodell zu beschleunigen, indem diese durch ein lineares Passagierflussmodell approximiert wird. Der Passagierfluss im Netzwerk wird dabei als ein lineares Mehrgüter-Flussproblem aufgefasst. Wir beschreiben zunächst das Modell des Passagierflusses außerhalb des Marktmodells und gehen danach auf die besonderen Integrationsaspekte ein.

6.4.2.1 Modell des Passagierflusses

Die Verteilung der Passagiere auf die Flüge des Flugnetzwerks kann mit einem Netzwerkfluss modelliert werden. Das Passenger Flow Modell (PFM) bildet den Passagierfluss im Netzwerk als ein Mehrgüter-Flussproblem ab. Die Güter entsprechen den Passagieren in den untersuchten Märkten (Reiseverbindungen zwischen zwei Städten). Die Kanten des Netzwerks werden von den Legs gebildet. Die Kapazitäten sind durch die Sitzplatzanzahl der eingesetzten Flugzeuge auf den Legs definiert. Gesucht ist ein gültiger Fluss im Netzwerk mit dem maximalen Gewinn als Zielfunktion. An dieser Stelle verweisen wir auf einen wichtigen Unterschied zu Modellen des Passagierverhaltens hin: die Passagiere entscheiden sich nicht für die für sie günstigste Verbindung. Es wird vielmehr angenommen, dass die Fluggesellschaft durch Steuerung der Buchungen die Passagiere auf die Verbindungen bringen kann, die den Gewinn der Gesellschaft maximieren.

Bei der Definition der Transportgüter können unterschiedliche Detaillierungsgrade gewählt werden. Für jeden Markt (Städtepaare) benötigt man mindestens ein Transportgut. Des Weiteren können zusätzlich die Reiseroute (itinerary), die Klasse (Economy, Business, First), die Buchungsklasse (fare) und/oder der Ort des Ticketverkaufs (point of sale) berücksichtigt werden.

Wir definieren an dieser Stelle das Passenger Flow Modell für die gröbste Detaillierungsstufe: auf ODI-Basis (origin, destination, itinerary). Dazu benötigen wir die folgenden Eingabedaten:

$\mathcal{L}$	Legs des Flugplans
$\mathcal{S}$	Menge der Flughäfen
OD	$\subseteq \mathcal{S}^2$ Menge aller Passagier-Märkte
d_{od}	Geschätzte Anzahl der Passagiere im Markt od
P^{od}	Mögliche Reiserouten für Passagiere aus dem Markt od
$\delta_{l,p}$	Indikator, der angibt, ob ein Leg l zur Route p gehört ($\delta_{l,p} = 1$) oder nicht ($\delta_{l,p} = 0$)
f^{od}	Erlöse pro Passagier aus dem Markt od
c_l	Kapazität (Sitzplatzanzahl) des Legs l

Für jede Reiseroute p eines Marktes od führen wir eine Variable x_p^{od} ein, die die Anzahl der Passagiere von Markt od auf Reiseroute p beschreibt. Dann lässt sich das ODI-Passagierflussmodell wie folgt aufstellen:

Modell 6.1 (Passenger Flow Modell (PFM)).

$$\textit{Maximiere} \sum_{od \in OD} \sum_{p \in P^{od}} f^{od} x_p^{od} \tag{6.1}$$

unter den Nebenbedingungen

$$\sum_{p \in P^{od}} x_p^{od} \leq d_{od} \qquad \forall od \in OD \tag{6.2}$$

$$\sum_{od \in OD} \sum_{p \in P^{od}} \delta_{l,p} x_p^{od} \leq c_l \qquad \forall l \in \mathcal{L} \tag{6.3}$$

$$x_p^{od} \geq 0 \qquad \forall od \in OD, p \in P^{od} \tag{6.4}$$

Die Zielfunktion (6.1) maximiert die Erlöse der Fluggesellschaft. Die Restriktionen (6.2) stellen sicher, dass auf allen möglichen Reiserouten, die den Markt od bedienen, nicht mehr Passagiere transportiert werden als geschätzt wurden. Die Ungleichungen (6.3) erzwingen, dass die Kapazität auf den Legs nicht überschritten wird. Die Entscheidungsvariablen x_p^{od} werden nicht als ganzzahlig vorausgesetzt, da es sich bei den Passagierzahlen um Schätzungen handelt. Wegen einer inhärenten Ungenauigkeit der Prognosen der Passagierzahlen wird an dieser Stelle auf die Ganzzahligkeit verzichtet und mit fraktionalen Flüssen gearbeitet.

Lineare Passagierflussmodelle werden in der Literatur für unterschiedliche Zwecke benutzt [Glover et al., 1982], [Phillips et al., 1991], [Farkas, 1996]. Neben einer Modellierung des Passagierflusses im Netzwerk werden ähnliche Modelle im Bereich des Ertragsmanagements dazu verwendet, Buchungsanfragen zu verarbeiten und über die Verfügbarkeit von Plätzen auf Flügen zu entscheiden [Boer et al., 2002], [Boyd, 2002], [Williamson, 1988] und [Williamson, 1992].

6.4.2.2 Beschreibung der Integrationsstrategie

Bei dieser Variante modellieren wir den Passagierfluss mit Hilfe des Passenger Flow Modells und koppeln diese neue Komponente an die Lokale Suche Heuristik. Abbildung 6.5 zeigt die Vorgehensweise.

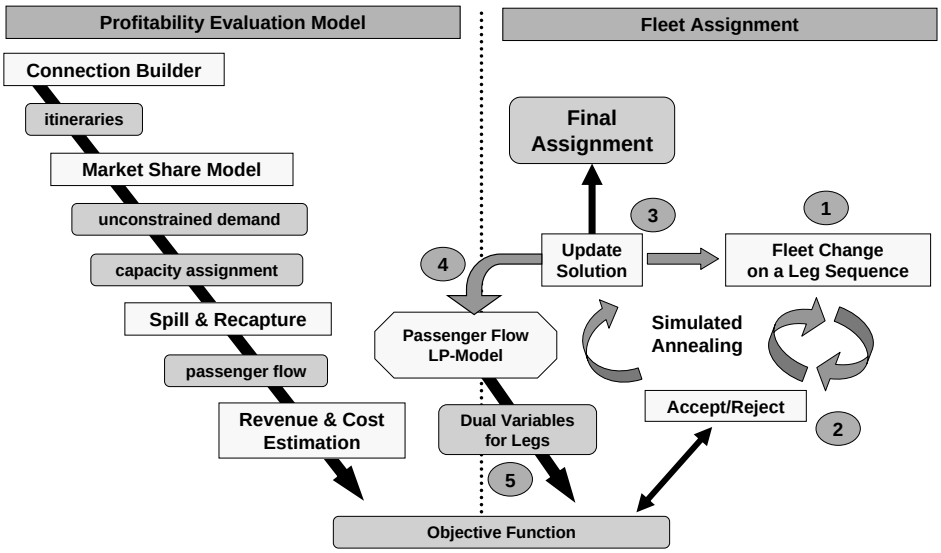


Abbildung 6.5: Passenger Flow Modell als Zwischenkomponente

Die Schritte bis zur ersten Lösungsübermittlung verlaufen wie bei der Integration des Marktmodells. Anstatt nun die aktuelle Flottenzuweisung an das Marktmodell zu schicken, verwenden wir das PFM, um eine aktualisierte Zielfunktion zu bestimmen. Das Passenger Flow Modell berechnet neben einem Passagierfluss zusätzliche Informationen und schickt diese als neue Zielfunktion an die Flottenzuweisung.

Die Kernidee besteht darin, für jede Netzwerkkante die dualen Variablen zu berechnen und diese der Flottenzuweisung zur Verfügung zu stellen. Die Nebenbedingungen (6.3) begrenzen den Fluss auf den Legs. Der Wert der dualen Variable gibt für ein Leg an, wie groß die Änderung der Zielfunktion bei einer Änderung der Kapazität um genau eine Einheit ist. Im Bereich des Ertragsmanagements werden die dualen Werte daher häufig auch als bid-prices verwendet.

Kanten, deren Kapazität vom Passagierfluss nicht komplett ausgeschöpft wurde, haben nach der Complementary-Slackness-Bedingung einen dualen Wert von Null. Der Simulated Annealing Algorithmus für die Flottenzuweisung bekommt die dualen Werte aller Kanten im Netzwerk und erhält dadurch die Information, welche Kanten einen Engpass darstellen. Die Zielfunktion wird entsprechend angepasst. Konkret wird der duale Wert bei den Koeffizienten der Flugzeugtypen aufaddiert, die eine höhere Kapazität besitzen als die aktuelle Zuweisung. Bei Flugzeugtypen mit geringerer Kapazität wird der Zielfunktionskoeffizient um den dualen Wert reduziert.

Die Änderung der Zielfunktion im Simulated Annealing Algorithmus soll bewirken, dass die Typzuweisung besser dem geschätzten Passagierfluss entspricht. Wie der Algorithmus auf die Anpassung der Zielfunktion reagiert, wird in Abschnitt 6.5 untersucht.

Der Vorteil der zweiten Integrationsstrategie im Vergleich zu der ersten ist die schnellere Anpassung der Zielfunktion durch das Passenger Flow Modell. Außerdem wird hier aktiv versucht, Engpässe im Netzwerk zu beheben. Dadurch werden insbesondere die Netzwerkeffekte

besser berücksichtigt. Die Kommunikationsstruktur lässt sich ebenfalls in Bezug auf Beginn, Häufigkeit und Anzahl der Kommunikationsrunden steuern.

6.4.3 Ertragsmanagement und Flottenzuweisung

Die letzte Integrationsstrategie verbindet die Systeme des Ertragsmanagements mit der Flottenzuweisung. Die Motivation kommt aus der Tatsache, dass circa drei Monate vor dem Start eines Legs bereits einige Buchungen im Ertragsmanagementsystem vorliegen und dadurch eine genauere Abschätzung der Gewinne als mit einem Marktmodell möglich wird. Die kurzfristige Änderungsplanung der Flottenzuweisung kann entscheidend von dieser besseren Gewinnschätzung profitieren.

Das im Abschnitt 6.2 vorgestellte Marktmodell berechnet ca. ein halbes Jahr vor dem Start der nächsten Flugplanperiode Prognosen zu den Passagierzahlen im Netzwerk. Diese Prognosen werden zur Schätzung der erzielbaren Erlöse herangezogen. Neben der Erlösrechnung berücksichtigt das Marktmodell auch die Kosten für die Ausführung des Flugplans, die hauptsächlich von den auf den Legs eingesetzten Flugzeugtypen abhängig sind. Die Zielsetzung der Flottenzuweisung besteht in der Maximierung der Gewinnfunktion unter Einhaltung sämtlicher operationeller Restriktionen.

Zur Schätzung der Erlöse geht das Verfahren von Reiserouten aus, die keine Unterscheidung nach Klassen (First, Business, Economy) beinhalten. Vielmehr wird ein Durchschnittsertrag für alle Klassen berechnet. Die Nachfrage-Prognosen werden lediglich als Mittelwerte verwendet, der Prognosefehler bleibt in diesem Fall unberücksichtigt.

Im O&D-Revenue-Management werden Steuerparameter zur Bestimmung der Verfügbarkeit von Flügen und zur Entscheidung von Buchungsanfragen berechnet. Im Gegensatz zur lokalen Steuerung berücksichtigt die O&D-Steuerung Verdrängungseffekte innerhalb des Flugnetzes (Netzwerkeffekte). Die Nachfrage-Prognose berücksichtigt nicht nur die Reiserouten, sondern auch die Buchungsklassen, die eine genauere Betrachtung als die Klassen (First, Business, Economy) erlaubt. Weiterhin werden auch Prognose-Fehler erfasst und in die Schätzung eingearbeitet.

Wie auch in den beiden bereits beschriebenen Integrationsstrategien erfolgt zunächst die initiale Bewertung des Ausgangsflugplans durch das Marktmodell. Die Kommunikation des Systems für die Flottenzuweisung erfolgt in diesem Fall mit dem O&D Revenue Management System (siehe Abbildung 6.6).

Zunächst lädt das Revenue Management System Prognosen (einschließlich Prognose-Fehler) aus der Forecast Datenbank. Der O&D Optimizer berechnet daraus mit Hilfe der iterative prorated EMSR-Methode die Steuerparameter (bid-prices) und übermittelt sie an die Gesamt-Erlös-Schätzung. Im letzten Schritt wird unter Berücksichtigung der aktuellen Flottenzuweisung eine Erlösschätzung ermittelt. Diese Information wird dem Flottenzuweisungssystem zur Verfügung gestellt. Die Kostenberechnung erfolgt weiterhin auf der Grundlage der vom Marktmodell ermittelten Daten.

Dieses Integrationsstrategie ist im Rahmen einer Vorstudie zusammen mit Lufthans Systems erarbeitet worden. Sie ist allerdings noch nicht in die Praxis umgesetzt.

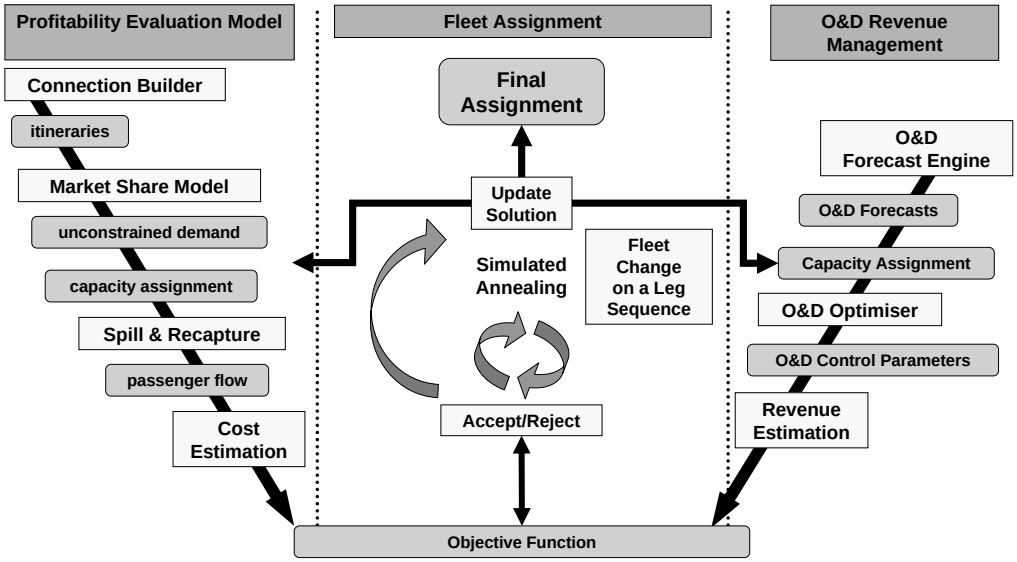


Abbildung 6.6: Einbeziehung von genaueren Erlösprognosen in der kurzfristigen Flottenzuweisung

6.5 Experimentelle Ergebnisse

Für die Experimente in diesem Abschnitt standen uns nur zwei brauchbare Datensätze zur Verfügung. Die Eingaben und Parameter eines Marktmodells sind sehr viel detaillierter als die Instanzen eines einfachen Flottenzuweisungsproblems und werden von Fluggesellschaften nicht nach draußen gegeben. Die Datensätze werden stellvertretend für eine Reihe von Szenarien benutzt, die bei mehreren Kunden von Lufthansa Systems ausgewertet wurden. Unsere Resultate spiegeln nach deren Aussagen das typische Verhalten der Integrationsstrategien wieder.

6.5.1 Datensätze

Zunächst beschreiben wir die zugrunde liegenden Netzwerke und die Passagierdaten, die für die Experimente benutzt wurden.

Anzahl	Datensatz A	Datensatz B
Legs	6287	9228
Flughäfen	97	160
Märkte	20358	6680
Itineraries (Reiseverbindungen)	34963	160635
Direkte itineraries	6740	14240
Single-connections (1 Zwischenstopp)	26573	140160
Double-connections (2 Zwischenstopps)	1650	5440

Der Datensatz A basiert auf einem kleineren Netzwerk als der Datensatz B, es wurden aber mehr Märkte berücksichtigt. Die Anzahl der Reiseverbindungen hängt stark von den Einstellungen des Marktmodells ab.

Damit sich Netzwerkeffekte auf die Gewinnberechnung auswirken, müssen Flottenzuweisungsinstanzen zwei Eigenschaften besitzen:

- Es muss Legs geben, mit denen mehr Fluggäste fliegen wollen als Sitzplätze in dem kleinsten Flugzeug, das das jeweilige Leg bedienen kann, verfügbar sind.

  Ansonsten könnte jede zulässige Flottenzuweisung immer alle Passagiere transportieren und die Erlöse wären konstant.

- Es muss Passagiere geben, deren Reiserouten aus mehr als einem Leg zusammengesetzt sind.

  Wenn auf einem Leg l für solch einen Passagier kein Platz im Flugzeug verfügbar ist, fliegt er keines der Legs der Reiseroute. Die Gesellschaft verliert also nicht nur die Erlöse des Passagiers für Leg l sondern für alle Legs der Reiseroute. Dies lässt sich aber mit der in der Flottenzuweisung verwendeten linearen Zielfunktion nicht modellieren.

Wir untersuchen nun die beiden Datensätze auf diese Eigenschaften. Bezeichne dazu

P	Menge aller möglichen Itineraries
P_c	Menge der Itineraries, die aus mehr als einem Leg bestehen
d_p	Unkapazitierte Anzahl an Passagieren auf der Itinerary p

Die *Passenger Connectivity Ratio*, gibt an, wie groß der Anteil der Passagiere ist, die über mehrere Zwischenstationen zu ihrem Zielflughafen reisen.

Definition 6.2 (Passenger Connectivity Ratio). *Das Passenger Connectivity Ratio ist das Verhältnis der Anzahl der Verbindungs-Passagiere zu der Gesamtanzahl der Passagiere im Netzwerk:*

$$PCR = \frac{\sum_{p \in P_c} d_p}{\sum_{p \in P} d_p}$$

Datensatz A besitzt eine Passenger Connectivity Ratio von 0.57, bei Datensatz B beläuft sie sich auf 0.36. In beiden Datensätzen sind also die Reiserouten vieler Passagiere aus mehreren Legs zusammengesetzt, so dass Netzwerkeffekte auftreten können.

Betrachtet man die Sitzplatzkapazitäten der möglichen Flugzeugtypen auf einer Flugstrecke und die geschätzte Anzahl der Passagiere auf dieser Flugstrecke, lassen sich drei Klassen von Legs definieren:

uncapacitated Die Nachfrage auf dem Leg liegt unterhalb der kleinsten Kapazität der möglichen Flugzeugtypen.

potentially capacitated Die Nachfrage auf dem Leg liegt oberhalb der kleinsten Kapazität der möglichen Flugzeugtypen, aber unterhalb der größten möglichen Kapazität auf der Strecke.

overcapacitated Die Nachfrage auf dem Leg liegt oberhalb der größten Kapazität der möglichen Flugzeugtypen.

Für unsere beiden Testinstanzen sieht die Verteilung auf die Klassen wie folgt aus:

Legs	Datensatz A	Datensatz B
uncapacitated	1312 (21%)	2835 (30%)
potentially capacitated	1588 (25%)	3805 (41%)
overcapacitated	3345 (54%)	2591 (29%)

Hier zeigt sich, dass ein Großteil der Legs der Passagiernachfrage nicht gewachsen sein werden. Es müssen also Passagiere zurückgewiesen werden, und beide Datensätze erfüllen die Voraussetzungen, das Netzwerkeffekte auftreten können. Datensatz A ist davon potentiell stärker betroffen, da er mehr überlastetet Legs enthält und auch mehr Passagiere, die Reiserouten mit mehr als einem Leg benutzen.

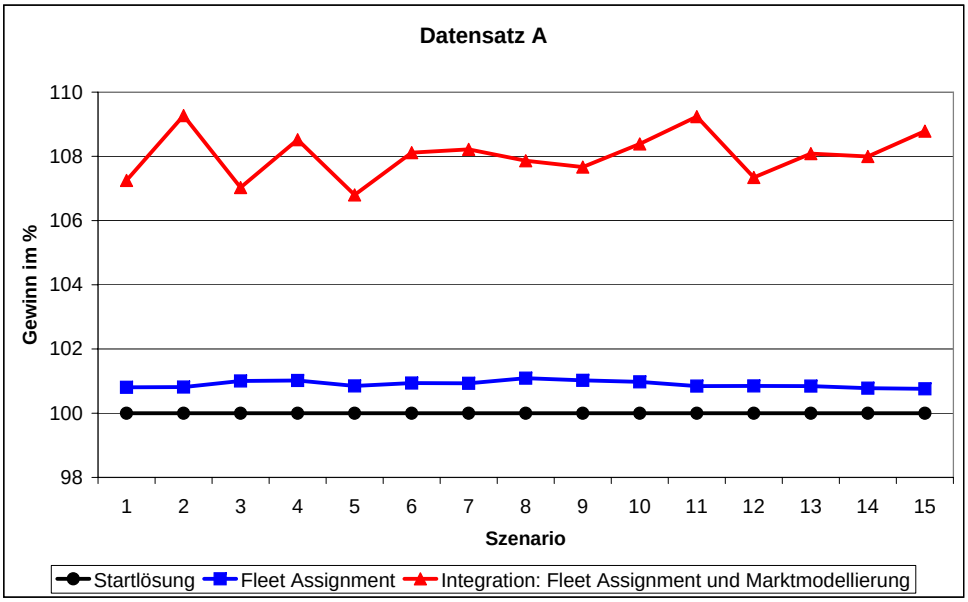
6.5.2 Ergebnisse der Integrationsstrategien

In der Abbildung 6.7 sind 15 Szenarien mit der ersten Integrationsstrategie dargestellt. Der Gewinnwert der Startlösung wird in diesem Experiment mit 100% angegeben. Das Ergebnis der Flottenzuweisung ohne die Interaktion mit dem Marktmodell (*Fleet Assignment*) erzielt im Durchschnitt eine Verbesserung von knapp einem Prozent gegenüber der Startlösung. Die in dieser Arbeit vorgestellte erste Integrationsstrategie erzielt eine durchschnittliche Verbesserung von 8%. Die absoluten Gewinnzahlen können auch hier nicht angegeben werden, weil sie eine vertrauliche Information der jeweiligen Fluggesellschaft darstellen.

Die zweite Integrationsstrategie implementiert das Modell des Passagierflusses als eine zusätzliche Komponente an der Schnittstelle zwischen der Marktmodellierung und der Flottenzuweisung. Das lineare Programm steuert die Zielfunktion der Flottenzuweisung mittels der dualen Werte der Kapazitätsrestriktionen der Flugstrecken (siehe Abschnitt 6.4.2).

In der Abbildung 6.8 ist das Konvergenzverhalten der Integration des Modells des Passagierflusses mit der Flottenzuweisung auf dem Datensatz B dargestellt. Im oberen Bild der Abbildung 6.8 sind die dualen Werte über die gesamte Laufzeit zu sehen. Bei den ersten Iterationen sind die dualen Werte relativ groß und steuern den Simulated Annealing Algorithmus entsprechend stark an. Zum Ende der Berechnung werden die dualen Werte deutlich kleiner. Die Kapazitätszuweisung im Simulated Annealing Algorithmus wurde an den vorhergesagten Fluss angepasst. Die Erhöhung der Kapazitäten auf den entsprechenden Flugstrecken würde keinen größeren Zugewinn mehr bedeuten.

Im unteren Teil der Abbildung 6.8 wurde die Differenz in der Zielfunktion des Simulated Annealing Algorithmus jeweils vor und nach einer Iteration mit dem Passagierflussmodell protokolliert. Auch hier ist die Konvergenz des Verfahrens deutlich zu sehen. Die Bewertung der aktuellen Lösung durch das System der Flottenzuweisung und das Modell des Passagierflusses nähern sich im Verlauf der Optimierung stark an.



			Integration:
Szenario	Startlösung	Fleet Assignment	Fleet Assignment und Marktmodellierung
1	100%	100.8%	107.3%
2	100%	100.8%	109.3%
3	100%	101.0%	107.0%
4	100%	101.0%	108.5%
5	100%	100.9%	106.8%
6	100%	100.9%	108.1%
7	100%	100.9%	108.2%
8	100%	101.1%	107.9%
9	100%	101.0%	107.7%
10	100%	101.0%	108.4%
11	100%	100.8%	109.2%
12	100%	100.8%	107.3%
13	100%	100.8%	108.1%
14	100%	100.8%	108.0%
15	100%	100.8%	108.8%
Mittelwert		100.9%	108.0%

Abbildung 6.7: Ergebnisse der ersten Integrationsstrategie

6.6 Zusammenfassung

In diesem Kapitel haben wir uns mit der Integration der Flottenzuweisung mit der Marktmodellierung bzw. dem Ertragsmanagement beschäftigt. Wir haben die beiden Planungsphase beschrieben und das durch sie erzielbare Verbesserungspotential für die Flottenzuweisung untersucht. In der Mittelfristplanung erfolgt die Integration mit der Marktmodellierung. Dadurch werden die im Flugnetz auftretenden Netzwerkeffekte besser berücksichtigt. In der Kurzfristplanung werden die wesentlich genaueren Passagierprognosen und Erlösschätzungen aus dem Ertragsmanagement benutzt, um die Kapazitäten durch veränderte Flottenzuweisung besser auf den Passagierfluss anzupassen.

Anhand der Experimente kann festgestellt werden, dass die Integration der Phasen der Marktmodellierung und der Flottenzuweisung erfolgreich durchgeführt worden ist. Die implementierte Strategie wird bei mehreren Fluggesellschaften im Produktionsbetrieb eingesetzt und

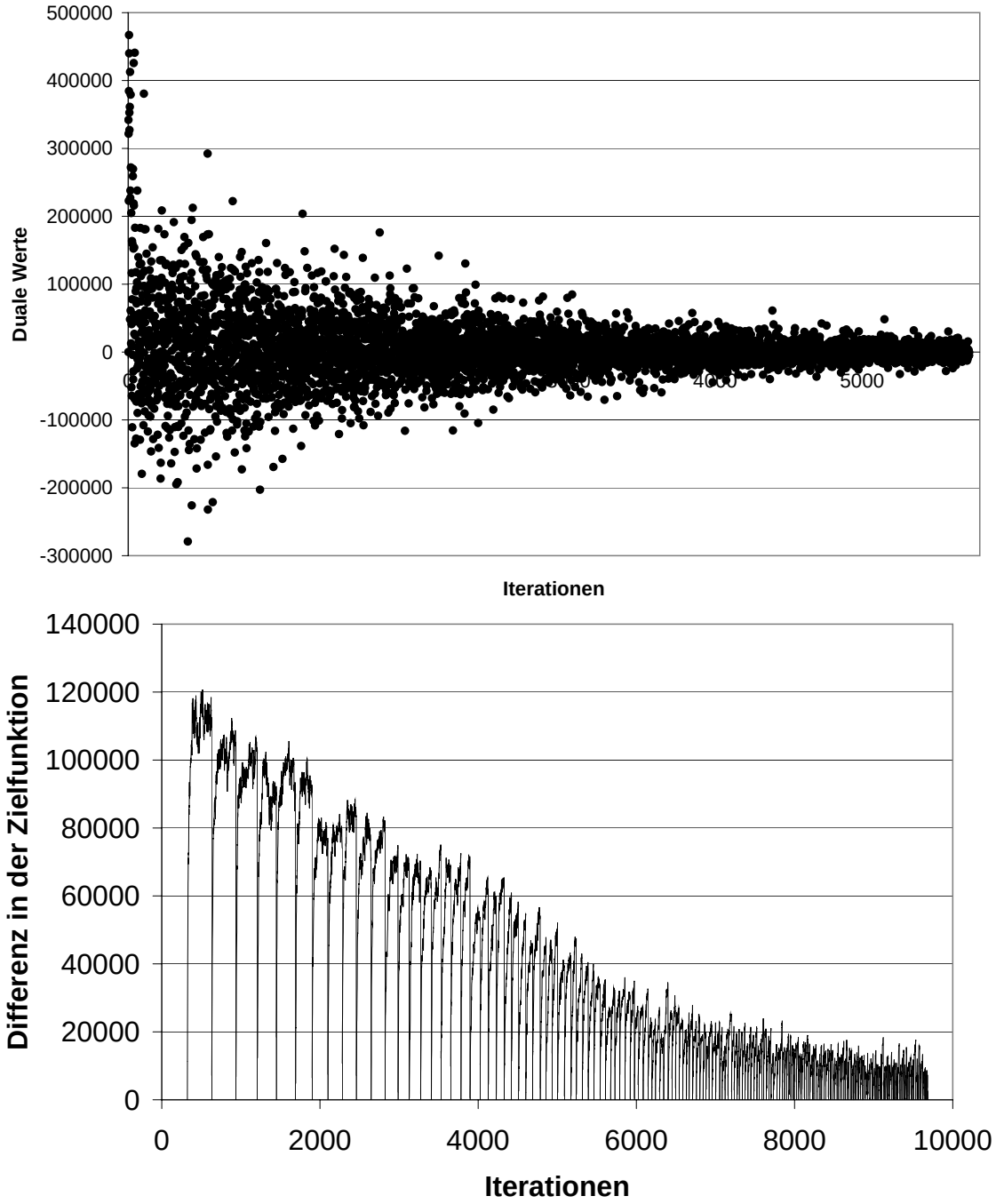


Abbildung 6.8: Konvergenz der dualen Werte und der Differenz in der Zielfunktion

leistet einen großen Beitrag zur Steigerung der Profitabilität der Flugpläne.

Die zweite Integrationsstrategie ist als Forschungsprototyp implementiert und mit realen Daten einer Fluggesellschaft getestet worden. Die Übernahme der Erkenntnisse in die Planungssysteme der Fluggesellschaften steht noch bevor.

Die dritte Integrationsstrategie wird in naher Zukunft im Rahmen einer umfangreicheren Studie evaluiert. Die hohe Aufwand für die Anpassung des Ertragsmanagement-Systems konnten

im Rahmen dieser Arbeit nicht geleistet werden. Eine Auswertung der Ergebnisse dieser Integrationsstrategie steht bevor. Die vorbereitenden Analysen seitens Lufthansa Systems lassen auf ein hohes Potential der vorgeschlagenen Vorgehensweise schließen.

— 7 —

Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Arbeit wurde das Problem der Flottenzuweisung in der Flugplanung behandelt. Es handelt sich dabei um eine der wichtigsten Planungsaufgaben bei der Erstellung eines Flugplans. Die in der Arbeit für das Flottenzuweisungsproblem entwickelten Algorithmen kommen in kommerziellen entscheidungsunterstützenden Systemen zum Einsatz, wo sie die Qualität der Flotteneinsatzplanung vieler Fluggesellschaften erfolgreich verbessern konnten.

Die Flottenzuweisung ist ein kombinatorisch schweres Optimierungsproblem, bei dem den vorgegebenen Flügen eines Flugplans die sie operierenden Flugzeugtypen zugewiesen werden müssen. Dabei muss sichergestellt werden, dass jedem Flug genau ein Flugzeugtyp zugewiesen wird und die benötigten Flugzeuge in der vorhandenen Flotte verfügbar sind. Aufgrund unterschiedlicher Sitzkapazitäten und operationeller Kosten der einzelnen Flugzeugtypen ist die Zielsetzung der Flottenzuweisung, eine gewinnmaximale, zulässige Lösung zu liefern.

Wir konnten neue Ergebnisse zur Komplexität des Flottenzuweisungsproblems zeigen. Insbesondere ergibt sich, dass das Flottenzuweisungsproblem bereits für zwei Flugzeugtypen streng NP-vollständig und nicht in polynomieller Zeit approximierbar ist. Des Weiteren haben wir die bekannten Ergebnisse vervollständigt, indem wir die Komplexität des azyklischen Flottenzuweisungsproblems und die Auswirkungen von Problemerweiterungen wie verbindungsabhängigen Gewinnen untersucht haben.

Zum Lösen des Flottenzuweisungsproblems haben wir neue exakte und heuristische Verfahren entwickelt. Die exakten Verfahren basieren auf IP-Formulierungen des Problems, die mit Standardlösern aus dem Bereich der ganzzahligen linearen Optimierung gelöst werden. Verschiedene neue Modelle für erweiterte Flottenzuweisungsprobleme sind entwickelt und vorgestellt worden. Die heuristischen Lösungsverfahren basieren auf der Lokale Suche-Idee, wobei hier die verwendete problemspezifische Nachbarschaft dafür verantwortlich ist, dass gute Lösungen in kurzer Zeit gefunden werden.

Durch die Berücksichtigung von stochastischen Eingabedaten, etwa für die Dauer eines Fluges, konnten wir störungsunempfindlichere Pläne generieren, die beim Auftreten von Verspätungen kostengünstig repariert werden können. Wir haben gezeigt, dass dadurch allerdings die Flottenzuweisung sogar PSPACE-vollständig wird. Durch die Beschreibung als ein Spiel gegen die Natur lässt sich das Problem aber zumindest heuristisch mittels Spielbaumsuche lösen, und experimentelle Ergebnisse haben gezeigt, dass selbst diese heuristischen Lösungen den Lösungen von Modellen, die nur mit nicht-stochastischen Eingabedaten arbeiten, überlegen sind.

Durch die Integration der Flottenzuweisung mit der Marktmodellierung und dem Ertragsmanagement konnten wir ferner die Bewertung von Zuweisungen erheblich verbessern, da die in der Flottenzuweisung zumeist angenommene lineare Gewinnfunktion nicht realitätsnah genug ist. Wir haben verschiedene Integrationsstrategien für unserer Lokale Suche Heuristiken beschrieben und gezeigt, dass sich dadurch der tatsächlich erzielbare Gewinn von Flottenzuweisungen deutlich steigern lässt.

Die Erkenntnisse dieser Arbeit führen zu neuen interessanten Fragestellungen und Anwendungsmöglichkeiten:

- Die aktuellen Lösungsverfahren für das Flottenzuweisungsproblem sind in der Lage, auch sehr große Probleminstanzen in verhältnismäßig kurzer Zeit fast optimal zu lösen.

  Somit ergibt sich Spielraum, das Flottenzuweisungsproblem um weitere praxisrelevante Anforderungen zu erweitern und damit stärker mit anderen Planungsaufgaben zu verknüpfen.

  Ein natürlicher Kandidat in diesem Zusammenhang ist das Rotation Building, bei dem aus der Lösung einer Flottenzuweisung Einsatzpläne für konkrete Flugzeuge erstellt werden. Eine Hauptschwierigkeit ist hier die Sicherstellung der erforderlichen Wartungsarbeiten.

  Ein weiterer Kandidat ist die Crewplanung. Die Crewplanung ist in hohem Maße von der Flotteneinsatzplanung abhängig und eine zumindest teilweise Berücksichtigung von komplizierten Anforderungen wie Ruhezeitregeln in der Flottenzuweisung würde zu deutlich verbesserten Crewplänen führen.

- Stochastische Eingabedaten kommen nicht nur in der Flottenzuweisung vor sondern treten in praktisch allen realen Planungsproblemen auf. Mit dem Reparaturspiel haben wie ein generisches Modellierungs- und Lösungsverfahren für mehrstufige Entscheidungsprobleme mit stochastischen Eingabedaten entwickelt.

  Der Einsatz der Reparaturspiels für andere Optimierungsprobleme aus den Bereichen Verkehrs- oder Produktionsplanung sollte auch hier zu robusteren, besseren Lösungen führen können.

- Das Potential der Integration von Ertragsmanagement und Flottenzuweisung ist noch nicht vollständig ausgeschöpft. Bereits durch die Verwendung der vergleichsweise groben Methoden der Marktmodellierung lassen sich bedeutend bessere Flottenzuweisungen produzieren. Das Ertragsmanagement sollte hier nochmals eine Verbesserung erzielen können, eine Umsetzung und Evaluierung der in dieser Arbeit vorgestellten Integrationsstrategie steht aber noch aus.

Literaturverzeichnis

[Abara, 1989] Abara, J. (1989). Applying integer linear programming to the fleet assignment problem. *Interfaces*, 19(4):20–28.

[Ahuja et al., 1993] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Prentice Hall, Englewood Cliffs, New Jersey.

[Althöfer, 1988] Althöfer, I. (1988). Root evaluation errors: How they arise and propagate. *ICCA Journal*, 11(3):55–63.

[Balas, 1998] Balas, E. (1998). Disjunctive programming: properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89:3–44. (ursprünglich MSRR# 348, Carnegie Mellon University, Juli 1974).

[Barnhart et al., 2003] Barnhart, C., Belobaba, P., and Odoni, A. R. (2003). Applications of operations research in the air transport industry. *Transportation Science*, 37(4):368–391.

[Barnhart et al., 1998] Barnhart, C., Boland, N., Clarke, L., Johnson, E., Nemhauser, G., and Shenoi, R. (1998). Flight string models for aircraft fleeting and routing. *Transportation Science*, 32(3):208–220.

[Belobaba, 1989] Belobaba, P. (1989). Application of a probabilistic decision model to airline seat inventory control. *Operations Research*, 37(2):183–197.

[Ben-Akiva and Bierlaire, 1985] Ben-Akiva, M. and Bierlaire, M. (1985). *Discrete choice analysis*. The MIT Press Series in Transportation Studies.

[Ben-Akiva and Bierlaire, 1999] Ben-Akiva, M. and Bierlaire, M. (1999). *Discrete Choice Methods and their Applications to Short Term Travel Decisions*. Handbook of Transportation Sciences.

[Berge and Hopperstad, 1993] Berge, M. and Hopperstad, C. (1993). Demand driven dispatch: A method for dynamic aircraft capacity assignment, models and algorithms. *Operations Research*, 41(1):153–168.

[Boer et al., 2002] Boer, S., Freling, R., and Piersma, N. (2002). Mathematical programming for network revenue management revisited. *European Journal of Operational Research*, 137:72–92.

[Borosh and Treybis, 1976] Borosh, I. and Treybis, L. (1976). Bounds on positive integral solutions of linear Diophantine equations. *Proc. American Mathematical Society*, 55:299–304.

[Boyd, 2002] Boyd, A. (2002). Revenue management and dynamic pricing. In *IMA Tutorial: Supply Chain and Logistics Optimization*.

[Cohn and Barnhart, 2003] Cohn, A. M. and Barnhart, C. (2003). Improving crew scheduling by incorporating key maintenance routing decisions. *Operations Research*, 51(3):387 –396.

[Cook, 1971] Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proc. 3rd ACM Symposium on Theory of Computing*, pages 151–158.

[Daniels and Kouvelis, 1995] Daniels, R. L. and Kouvelis, P. (1995). Robust scheduling to hedge against processing time uncertainty in sigle-stage production. *Management Science*, 41:363–376.

[Daskin and Panayotopoulos, 1989] Daskin, M. and Panayotopoulos, N. (1989). A Lagrangian relaxation approach to assigning aircraft to routes in hub and spoke networks. *Transportation Science*, 23(2):91–99.

[Desaulniers and Desrosiers, 1997] Desaulniers, G. and Desrosiers, J. (1997). Daily aircraft routing and scheduling. *Management Science*, 43(6):841–855.

[Donninger et al., 2004] Donninger, C., Kure, A., and Lorenz, U. (2004). Parallel Brutus: The First Distributed, FPGA Accelerated Chess Program. In *Proceedings of 18th International Parallel & Distributed Processing Symposium (IPDPS'04)*, pages 44 + CD ROM, Santa Fe. ACM.

[Ehrhoff et al., 2003] Ehrhoff, J., Grothklags, S., Halbsgut, J., Lorenz, U., and Sauerwald, T. (2003). Robust plans and disruption management in aircraft scheduling with the help of game tree search. In *Proc. 43rd Annual Symposium of the Airline Group of the International Federation of Operational Research Societies (AGIFORS 2003)*, Paris, France.

[Ehrhoff et al., 2005a] Ehrhoff, J., Grothklags, S., and Lorenz, U. (2005a). Das Reparaturspiel als Formalisierung von Planung unter Zufallseinflüssen, angewendet in der Flugplanung. In Günther, H.-O., Mattfeld, D. C., and Suhl, L., editors, *Supply Chain Management und Logistik: Optimierung, Simulation, Decision Support*, pages 337–358. Physica Verlag, Heidelberg.

[Ehrhoff et al., 2005b] Ehrhoff, J., Grothklags, S., and Lorenz, U. (2005b). Parallelism for perturbation management and robust plans. In *Proc. 11th International Euro-Par Conference (EUROPAR 2005)*, volume 3648 of *LNCS*, pages 1265–1274, Lisbon, Portugal.

[Engell et al., 2001] Engell, S., Märkert, A., Sand, G., and Schultz, R. (2001). Production planning in a multiproduct batch plant under uncertainty. *Preprint 495-2001, FB Mathematik, Gerhard-Mercator-Universität Duisburg*.

[Etschmaier and Mathaisel, 1984] Etschmaier, M. and Mathaisel, D. (1984). Aircraft scheduling: The state of the art. In *AGIFORS Symposium*.

[Even et al., 1976] Even, S., Itai, A., and Shamir, A. (1976). On the complexity of timetables and mutlticommodity flow problems. *SIAM Journal on Computing*, 5:691–703.

[Even and Tarjan, 1975] Even, S. and Tarjan, R. E. (1975). Network flow and testing graph connectivity. *SIAM Journal on Computing*, 4:507–518.

[Fahle et al., 2003] Fahle, T., Feldmann, R., Götz, S., Grothklags, S., and Monien, B. (2003). The aircraft sequencing problem. In Klein, R., Six, H.-W., and Wegner, L. M., editors, *Computer Science in Perspective*, volume 2598 of *LNCS*, pages 152–166. Springer.

[Farkas, 1996] Farkas, A. (1996). *The influence of network effects and yield management on airline fleet assignment decisions*. PhD thesis, MIT. Supervised by P. Belobaba.

[Ferguson and Dantzig, 1956] Ferguson, A. R. and Dantzig, G. B. (1956). The allocation of aircraft to routes - an example of linear programming under uncertain demand. *Management Science*, 3.

[Gertsbach and Gurevich, 1977] Gertsbach, I. and Gurevich, Y. (1977). Constructing an optimal fleet for a transportation schedule. *Transportation Science*, 11(1):20–36.

[Glover et al., 1982] Glover, F., Glover, R., Lorenzo, J., and McMillan, C. (1982). The passenger-mix problem in the scheduled airlines. *Interfaces*, 12(3):73–80.

[Götz et al., 1999] Götz, S., Grothklags, S., Kliewer, G., and Tschöke, S. (1999). Solving the weekly fleet assignment problem for large airlines. In *Proceedings of the Third Metaheuristics International Conference (MIC 1999)*, pages 241–246, Angra dos Reis, Brasil.

[Grothklags, 2000] Grothklags, S. (2000). Simulated Annealing für das Fleet Assignment Problem. Diplomarbeit, Universität Paderborn.

[Grothklags, 2003] Grothklags, S. (2003). Fleet assignment with connection dependent ground times. In Battista, G. D. and Zwick, U., editors, *Proc. 11th Annual European Symposium on Algorithms (ESA 2003)*, volume 2832 of *LNCS*, pages 667–678, Budapest, Hungary.

[Grothklags et al., 2002] Grothklags, S., Kliewer, G., and Weber, K. (2002). Improving revenue by system integration and co-operative optimization. In *Proc. 42nd Annual Symposium of the Airline Group of the International Federation of Operational Research Societies (AGIFORS 2002)*, Honolulu, USA.

[Grothklags et al., 2004] Grothklags, S., Lorenz, U., and Sauerwald, T. (2004). Experiments with the repair game. In *Aviation Application Cluster at Institute for Operations Research and Management Science (INFORMS)*, Denver, USA.

[Grothklags et al., 2006a] Grothklags, S., Lorenz, U., and Sauerwald, T. (2006a). On the computational complexity of multi-stage decision making under uncertainty for aircraft scheduling. In *Operations Research (OR 2006)*, Karlsruhe, Germany.

[Grothklags et al., 2006b] Grothklags, S., Lorenz, U., and Sauerwald, T. (2006b). Stochastic airline fleet assignment is PSPACE-complete. In *5th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2006)*, Lambrecht, Germany.

[Gu et al., 1994] Gu, Z., Johnson, E., Nemhauser, G., and Wang, Y. (1994). Some properties of the fleet assignment problem. *Operations Research Letters*, 15:59–71.

[Hájek, 1985] Hájek, B. (1985). A tutorial survey of the theory and applications of simulates annealing. In *Proc. 24th IEEE Conference on Decision and Control*, pages 755–760.

[Hájek, 1988] Hájek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13:311–329.

[Hane et al., 1995] Hane, C., Barnhart, C., Johnson, E., Marsten, R., Nemhauser, G., and Sigismondi, G. (1995). The fleet assignment problem: solving a large-scale integer program. *Mathematical Programming*, 70:211–232.

[Hopcroft and Karp, 1973] Hopcroft, J. E. and Karp, R. M. (1973). A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2:225–231.

[Horvitz et al., 1988] Horvitz, E., Breese, J., and Henrion, M. (1988). Decision theory in expert systems and artificial intelligence. *Journal of Approximate Reasoning, Special Issue on Uncertainty in Artificial Intelligence*, pages 247–302.

[Jacobs and Günther, 2000] Jacobs, T. L. and Günther, D. (2000). Benefits and barriers associated with the integration of airline pricing, yield management and scheduling decisions. In *Proceedings of the 40th Annual Symposium of the Airline Group of the International Federation of Operational Research Societies (AGIFORS)*.

[Jacobs et al., 2000] Jacobs, T. L., Ratliff, R., and Smith, B. C. (2000). Soaring with synchronized systems. *OR/MS Today*, pages 36–44.

[Jarrah et al., 2000] Jarrah, A., Goodstein, J., and Narasimhan, R. (2000). An efficient airline re-fleeting model for the incremental modification of planned fleet assignments. *Transportation Science*, 34(4):349–363.

[Karp, 1972] Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York.

[Klabjan, 2005] Klabjan, D. (2005). Large-scale models in the airline industry. In Desaulniers, G., Desrosiers, J., and Solomon, M., editors, *Column Generation*. Kluwer Academic Publishers. to appear.

[Klabjan et al., 2002] Klabjan, D., Johnson, E., Nemhauser, G., Gelman, E., and Ramaswamy, S. (2002). Airline crew scheduling with time windows and plane count constraints. *Transportation Science*, 36:337–348.

[Kliewer, 2005] Kliewer, G. (2005). *Optimierung in der Flugplanung: Netzwerkentwurf und Flottenzuweisung*. Dissertation, Universität Paderborn.

[Kniker, 1998] Kniker, T. (1998). *Itinerary-based airline fleet assignment*. PhD thesis, MIT. Supervised by C. Barnhart.

[Knuth and Moore, 1975] Knuth, D. and Moore, R. (1975). An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326.

[Koppelman and Sethi, 2000] Koppelman, F. and Sethi, V. (2000). *Closed Form Discrete Choice Models*. Handbook of Transport Modeling. Pergamon Press.

[Kouvelis et al., 2000] Kouvelis, P., Daniels, R. L., and Vairaktarakis, G. (2000). Robust scheduling of a two-machine flow shop with uncertain processing times. *IIE Transactions*, 32(5):421–432.

[Kuhn, 1955] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quaterly*, 2:83–97.

[Lefeld and Pölt, 1995] Lefeld, M. and Pölt, S. (1995). Standardwochen Marktmodell für OPNET. Analyse des SPEED-Marktmodells. Technical report, Lufthansa Systems.

[Leon et al., 1994] Leon, V. J., Wu, S. D., and Storer, R. H. (1994). A game-theoretic control approach for job shops in the presence of disruptions. *International Journal of Production Research*, 32(6):1451–1476.

[Li et al., 2006] Li, D., Huang, H.-C., Morton, A. D., and Chew, E.-P. (2006). Simultaneous fleet assignment and cargo routing using Benders decomposition. *OR Spectrum*, 28:319–335.

[Littlewood, 1972] Littlewood, K. (1972). Forecasting ans control of passenger bookings. In *AGIFORS Symposium*.

[Lohatepanont, 2002] Lohatepanont, M. (2002). *Airline fleet assignment and schedule design: integrated models and algorithms*. PhD thesis, MIT. Supervised by C. Barnhart.

[Lorenz and Monien, 2004] Lorenz, U. and Monien, B. (2004). Error analysis in minimax trees. *Journal of Theoretical Computer Science*, 313:485–498.

[LufthansaSystems, 1997] LufthansaSystems (1997). SPEED-V: Technische Dokumentation. Technical report, Lufthansa Systems.

[Maurer, 2003] Maurer, P. (2003). *Luftverkehrsmanagement*. Oldenbourg Verlag.

[Müller-Bungart, 2003] Müller-Bungart, M. (2003). *Prognose der Passagiernachfrage im Linienluftverkehr*. Diplomarbeit, Universität zu Köln.

[Mörhing et al., 1999] Mörhing, R. H., Schulz, A. S., and Uetz, M. (1999). Approximation in stochastic scheduling: The power of LP-based priority scheduling. *Journal of ACM*, 46:924–942.

[Mulvey et al., 1995] Mulvey, J. M., Vanderbei, R. J., and Zenios, S. A. (1995). Robust optimization of large-scale systems. *Operations Research*, 43:264–281.

[Nau, 1979] Nau, D. (1979). *Quality of Decision versus depth of search on game trees*. PhD thesis, Duke University, Durham, NC.

[Nitschke, 1997] Nitschke, T. (1997). *Dynamic Fleet Assignment*. Diplomarbeit, Martin-Luther-University Halle-Wittenberg, Lufthansa Systems.

[Osman, 1993] Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451.

[Papadimitriou, 1985] Papadimitriou, C. H. (1985). Games against Nature. *Journal of Computer and System Science*, 31:288–301.

[PARALOR, 1997] PARALOR (1997). PARALOR: Gemeinsamer Abschlußbericht. Technical report. PARALOR: Parallel Algorithms for Large Scale Operations Research Problems (BMBF Projekt).

[Phillips et al., 1991] Phillips, R., Boyd, D., and Jr., T. (1991). An algorithm for calculating consistent itinerary flows. *Transportation Science*, 25(3):225–239.

[Radicke, 1994] Radicke, U.-D. (1994). *Algorithmen für das Fleet Assignment von Flugplänen. Optimierung auf Marktmodellbasis*. Verlag Shaker.

[Reinefeld, 1983] Reinefeld, A. (1983). An Improvement of the Scout Tree Search Algorithm. *ICCA Journal*, 6(4):4–14.

[Rexing, 1997] Rexing, B. (1997). *Airline Fleet Assignment with time windows*. Master, MIT. Supervised by Barnhart.

[Römisch and Schultz, 2001] Römisch, W. and Schultz, R. (2001). Multistage stochastic integer programming: an introduction. In Grötschel, M., Krumke, S. O., and Rambau, J., editors, *Online Optimization of Large Scale Systems*, pages 579–598. Springer.

[Rushmeier and Kontogiorgis, 1997] Rushmeier, R. and Kontogiorgis, S. (1997). Advances in the optimization of airline fleet assignment. *Transportaiton Science*, 31(2):159–169.

[Scheidler, 2003] Scheidler, M. (2003). *Discrete Choice Models for Airline Network Management*. PhD thesis, Universität Frankfurt.

[Scholl, 2001] Scholl, A. (2001). *Robuste Planung und Optimierung: Grundlagen, Konzepte und Methoden, Experimentelle Untersuchungen*. Springer.

[Sharma et al., 2000] Sharma, D., Ahuja, R. K., and Orlin, J. B. (2000). Neighborhood search algorithms for the combined through-fleet assignment model. Talk at 17th International Symposium on Mathematical Programming (ISMP'00).

[Shenoi, 1996] Shenoi, R. (1996). *Integrated Airline Schedule Optimization: Models and solution methods*. PhD thesis, MIT Cambridge. Supervised by Barnhart.

[Shmoys and Swamy, 2004] Shmoys, D. B. and Swamy, C. (2004). Stochastic optimization is (almost) as easy as deterministic optimization. In *Proc. 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 229–237.

[Sieber, 1995] Sieber, G. (1995). Spill and recapture user guide. Technical report, Lufthansa Systems.

[Sosnowska, 1999] Sosnowska, D. (1999). Optimization of a simplified fleet assignment problem with metaheuristics: Simulated annealing and GRASP. In *Approximation and complexity in numerical optimization: continuous and discrete problems*. Kluwer.

[Sosnowska and Rolim, 2000] Sosnowska, D. and Rolim, J. (2000). Fleet scheduling optimization: A simulated annealing approach. In Burke, E. and Erben, W., editors, *Practice and Theory of Automated Timetabling III (PATAT 2000)*. Springer-Verlag Heidelberg.

[Soumis et al., 1980] Soumis, F., Ferland, J.-A., and Rousseau, J. (1980). A model for large-scale aircraft routing and scheduling problems. *Transportation Research B*, 14B(1/2):191–201.

[Sterzenbach and Conrady, 2003] Sterzenbach, R. and Conrady, R. (2003). *Luftverkehr*. Oldenbourg Verlag.

[Stockmeyer, 1976] Stockmeyer, L. J. (1976). The polynomial time hierarchy. *Theoretical Computer Science*, 1:1–22.

[Subramanian et al., 1994] Subramanian, R., Sheff, R., Quillinan, J., Wiper, D., and Marsten, R. (1994). Coldstart: Fleet assignment at Delta Air Lines. *Interfaces*, 24(1):104–120.

[Suhl, 1995] Suhl, L. (1995). *Computer-aided scheduling: an airline perspective*. Gabler Edition Wissenschaft. Berlin, Tech. Univ., Habil.-Schr.,1993.

[Talluri, 1996] Talluri, K. (1996). Swapping applications in a daily airline fleet assignment. *Transportation Science*, 30(3):237–248.

[Talluri and van Ryzin, 1998] Talluri, K. T. and van Ryzin, G. J. (1998). An analysis of bid-price controls for network revenue management. *Manage. Sci.*, 44(11):1577–1593.

[Talluri and van Ryzin, 2005] Talluri, K. T. and van Ryzin, G. J. (2005). *The Theory and Practice of Revenue Management*, volume 68 of *International Series in Operations Research and Management Science*. Springer.

[Weber et al., 2003] Weber, K., Sun, J., Sun, Z., Kliewer, G., Grothklags, S., and Jung, N. (2003). Systems integration for revenue-creating control processes. *Journal of Revenue and Pricing Management*, 2:120–137.

[Williamson, 1988] Williamson, E. (1988). *Comparison of Optimization Techniques for Origin-Destination Seat Inventory Control*. Master thesis, MIT. Supervised by Simpson.

[Williamson, 1992] Williamson, E. (1992). *Airline Network Seat Inventory Control: Methodologies and Revenue Impacts*. PhD thesis, MIT. Supervised by P. Belobaba.

[Yan and Young, 1996] Yan, S. and Young, H.-F. (1996). A decision support framework for multi-fleet routing and multi-stop flight scheduling. *Transportation research A*, 30(5):379–398.

[Yu and Thengvall, 2002] Yu, G. and Thengvall, B. G. (2002). *Handbook of applied optimization*, chapter Airline optimization, pages 689–703. Oxford University Press.

[Yu and Yang, 1998] Yu, G. and Yang, J. (1998). *Handbook of combinatorial optimization*, chapter Optimization applications in the airline industry, pages 635–726. Kluwer Academic Publishers.