Langfristige Schulungsplanung für Piloten mittels ganzzahliger Optimierung

D I P L O M A R B E I T

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Mathematik

eingereicht von Alexander Tschakert,
geboren am 1981-08-17 in Zeulenroda

Betreuer: Prof. Dr. Christoph Helmberg

Chemnitz, 2004-07-01

Abstract

Zur Flugbedarfsdeckung durch und zur Karriereplanung von Piloten bieten Fluggesellschaften Schulungsprogramme zur Umschulung auf höherwertige Cockpitfunktionen und Flugzeugtypen an. In der vorliegenden Arbeit wird in Zusammenarbeit mit der Lufthansa Systems Airline Services GmbH ein gemischt ganzzahliges Optimierungsmodell zur langfristigen Schulungsplanung erstellt. Die Vor- und Nachteile des gewählten Modells werden diskutiert und es wird bewiesen, dass das zugehörige Entscheidungsproblem in der Komplexitätsklasse der NP-vollständigen Probleme liegt. Als Lösungsverfahren für das gemischt ganzzahlige Problem wird der Branch&Cut-Algorithmus vorgestellt. Darüber hinaus wird zur Erzeugung guter ganzzahliger Lösungen eine Heuristik mit (theoretisch) polynomieller Laufzeit entwickelt, die auf der Lösung der linearen Relaxierung des Problems aufbaut. Mit Hilfe einer Implementierung der entworfenen Algorithmen und Lösungsverfahren werden deren Qualität, Laufzeit sowie Vor- und Nachteile anhand praktischer Daten, die in Zusammenarbeit mit der Lufthansa Systems Airline Services GmbH erhoben wurden, untersucht.

Vorwort

Die folgende Arbeit behandelt die Problemstellung der langfristigen Schulungsplanung für Piloten und ist in 5 Kapitel unterteilt. In Kapitel 1 werden die zu untersuchende Problemstellung verbal formuliert und alle für das Verständnis des Problems wichtigen Fachbegriffe definiert und erläutert. Ein Glossar mit Kurzbeschreibungen aller Fachbegriffe ist in Anhang A zu finden. In Kapitel 2 wird die Problemstellung als ein gemischt ganzzahliges Optimierungsproblem mathematisch formuliert. Mit Kapitel 3 weisen wir dann nach, dass das aufgestellte Problem in der Komplexitätsklasse der NP-vollständigen Probleme liegt. Als Lösungsverfahren wird in Kapitel 4 der Branch&Cut-Algorithmus vorgestellt und aufgrund des Komplexitätsresultats aus Kapitel 3 eine Heuristik zur Konstruktion guter Schulungspläne entwickelt. Die vorgestellten Algorithmen werden in englischer Sprache angegeben, um eine gute Vergleichbarkeit zur in Englisch geschriebenen Implementierung zu gewährleisten. Eine ebenfalls in dieser Arbeit aus dem Englischen übernommene Schreibweise ist die Verwendung eines Kommas statt eines Punktes in der Dezimaldarstellung reeller Zahlen, um Konfusionen mit grammatikalischen Kommata in Sätzen zu vermeiden. Kapitel 4 endet mit der Diskussion numerischer Ergebnisse der besprochenen Algorithmen anhand realer Eingangsdaten. Kapitel 5 schließlich behandelt die Umwandlung von in Kapitel 4 erstellten Schulungsplänen in eine für Lufthansa praktikable Form. Die in der vorliegenden Arbeit verwendeten standardisierten mathematischen Schreibweisen seien im Folgenden kurz zusammengefasst. Wie üblich bezeichne

- $\mathbb{R}$ die Menge der reellen Zahlen, $\mathbb{R}_+$ die Menge der positiven reellen Zahlen (≥ 0),
- $\mathbb{Q}_+$ die Menge der positiven rationalen Zahlen (≥ 0),
- $\mathbb{Z}$ die Menge der ganzen Zahlen,
- $\mathbb{N}$ die Menge der natürlichen Zahlen, $\mathbb{N}_0$ die Menge der natürlichen Zahlen einschließlich 0.

Das euklidische Skalarprodukt zweier Vektoren $c, x \in \mathbb{R}^n$ werden wir als $cx := \sum_{i=1}^{n} c_i x_i$ schreiben. Die Funktion $\lfloor\ \rfloor : \mathbb{R} \to \mathbb{Z}, x \mapsto \max\{z \in \mathbb{Z} : z \leq x\}$ ordnet jeder reellen Zahl die größte ganze Zahl zu, die nicht größer als sie ist (floor-Funktion). Analog ordnet $\lceil\ \rceil : \mathbb{R} \to \mathbb{Z}, x \mapsto \min\{z \in \mathbb{Z} : z \geq x\}$ jeder reellen Zahl die kleinste ganze Zahl zu, die nicht kleiner als diese Zahl ist (ceiling-Funktion). Für eine beliebige Menge J nehme die Indikatorfunktion χ_J genau für die Elemente aus J den Wert 1 und ansonsten den Wert 0 an. Beispiele kennzeichnen wir im Text durch das Symbol ■.

■ $\binom{11}{8}\binom{-1}{20} = 149,\ \lfloor 5.7 \rfloor = 5,\ \lfloor 3 \rfloor = 3,\ \lceil 1.6 \rceil = 2,\ \lceil -8.9 \rceil = -8,\ \chi_{\{3,5\}}(5) = 1,\ \chi_{\mathbb{Z}}\left(\frac{1}{2}\right) = 0$

Es sei darauf hingewiesen, dass sich das Verwenden der Wir-Form in der vorliegenden Arbeit nicht auf die Existenz mehrerer Autoren bezieht, sondern der Leser damit zur kooperativen Erarbeitung des Textes mit dem Autor eingeladen werden soll.

Ich danke meinem Betreuer Prof. Dr. Helmberg für sein jederzeit offenes Ohr und die freundliche und kompetente Unterstützung. Danken möchte ich außerdem den Programmierern von Lufthansa Systems für die konstruktive und sehr angenehme Zusammenarbeit. Mein besonderer Dank gilt schließlich meinen Eltern, die mir in vielen Arbeitsstunden den Rücken frei gehalten und mich liebevoll unterstützt haben.

Chemnitz, 2004-07-01 Alexander Tschakert

Inhaltsverzeichnis

Kapitel 1

Problemstellung

Diese Arbeit hat zum Ziel, eine automatische Schulungsplanung für Piloten zu entwickeln. Jene Problemstellung wird im Weiteren mit **SPTS** (Strategic Pilot Training Scheduling) bezeichnet. Im folgenden Kapitel werden wir alle wichtigen Fachbegriffe einführen und die zu untersuchende Problemstellung SPTS verbal formulieren.

1.1 Planungseinheiten

Fluggesellschaften haben mit ihren Flugzeugen meist eine Vielzahl an Bedürfnissen abzudecken. Verschiedene Anforderungen an die zu leistende Streckenlänge (Kurz-/Langstrecke) und an Fracht- und Passagierkapazitäten ergeben die Notwendigkeit verschiedener **Flugzeug-Typen** (**Muster**). Umgekehrt wird ein Flugzeug-Typ oft von mehreren Fluggesellschaften genutzt.

Eine **Flotte** ist gegeben durch

(1) die Fluggesellschaft (Airline) und

(2) den Flugzeug-Typ (Aircraft).

Als Beispiele seien DLH-B737 (die Flotte der Boeing737 der Deutschen Lufthansa), CFG-A320 (die Flotte der Airbus320 der Condor Fluggesellschaft) und DLH-A320 genannt.

Jeder Pilot gehört zu jedem Zeitpunkt seiner Karriere zu genau einer Flotte. Innerhalb der Flotten werden Piloten zusätzlich in **Planungseinheiten** (Planning Units, **PU**s) eingeteilt, abhängig von

(3) der **Cockpitfunktion**, z.B. Captain (CP), First Officer (FO), Senior First Officer (SFO),

(4) der **Homebase** (Flughafen, bei dem der Pilot stationiert ist), z.B. Frankfurt, München,

(5) der **Subfunktion** (Auswahl spezieller Qualifikationen).

Für die langfristige Schulungsplanung SPTS wird eine Einteilung anhand der zwei letztgenannten Punkte nicht nötig sein, wir werden also PUs nur durch die Punkte (1) bis (3) charakterisieren. Zur Bezeichnung von PUs benutzen wir die Schreibweise *Airline-Aircraft-Cockpitfunktion*:

■ Airline	DLH			
Flotte	DLH-A320		DLH-B737	
PU	DLH-A320-CP	DLH-A320-FO	DLH-B737-CP	DLH-B737-FO

Die Anzahl der bei Lufthansa existierenden Planungseinheiten beläuft sich auf ungefähr 25.

1.2 Schulungen

Um den Flugprogrammbedarf einer PU zu decken, müssen zu jedem Zeitpunkt genügend Piloten vorhanden sein. Da sowohl der Flugprogrammbedarf durch saisonale Einflüsse wie Klima und Urlaubszeiten stark schwankt als auch der Pilotenbestand durch Altersabgänge und Kündigungen langsam sinkt, besteht die Notwendigkeit, neue Piloten auszubilden und Piloten zwischen den PUs umzuschulen.

Die Ausbildung neuer Piloten der Lufthansa erfolgt in der Verkehrsfliegerschule in Bremen, wo die angehenden Piloten (**NFF**s[1]) eine 3-jährige Ausbildung durchlaufen. Danach können sie sich für bestimmte PUs, sogenannte **Einstiegsmuster**, bewerben. Dies sind die First Officer-PUs der Kurzstrecken-Flotten, da Neulinge verständlicherweise noch nicht als Captains oder in passagierstarken Langstrecken-Flugzeugen eingesetzt werden.

Nach einer vorgeschriebenen Mindestverweildauer auf einem Einstiegsmuster können sich die Piloten für andere PUs bewerben. Generell gibt es nur eine gewisse Anzahl **zulässiger Übergänge** zwischen den PUs. Damit wird – wie bei den Einstiegsmustern – bezweckt, dass beim Wechsel auf eine andere Flotte zunächst als FO und dann erst als CP gearbeitet wird. Für einen Übergang von PU p zu PU q heißt p **Stamm-PU** und q **Ziel-PU** des Übergangs.
Ein Beispiel einer typischen DLH-Pilotenkarriere ist:

$$\text{A320-FO} \rightarrow \text{A340-FO} \rightarrow \text{A340-SFO} \rightarrow \text{A320-CP} \rightarrow \text{A340-CP}.$$

Um zwischen zwei PUs zu wechseln, nehmen die Piloten an Kursen teil. Ein **fixierter Kurs** ist durch drei Faktoren festgelegt:

- den Startzeitpunkt (Wann?)
- die Ziel-PU (Wohin?)
- die Anzahl der Teilnehmer (Wie viele?).

Ein Kurs qualifiziert die teilnehmenden Piloten (**Trainees**) innerhalb von drei Phasen (Theorie, Simulatortraining und Linienflug) fachlich auf die Ziel-PU, weitestgehend unabhängig von den Vorkenntnissen und den bereits durchlaufenen PUs. Deshalb können an einem Kurs problemlos Piloten verschiedener Stamm-PUs teilnehmen – entscheidend ist einzig die gemeinsame Ziel-PU. Sind zusätzlich

- die Stamm-PUs der Trainees (Woher?)

gegeben, so sprechen wir von einem **virtuellen Kurs**. Die Bezeichnung rührt daher, dass noch keine konkreten Piloten für den Kurs eingeteilt wurden, sondern nur Platzhalter für die Übergänge (**virtuelle Trainees**). Werden schließlich

- die realen Piloten (Wer?)

für den Kurs festgeschrieben, so wird er zu einem **realen Kurs**. Die Teilnahme eines Trainees an einem realen Kurs wird **Type-Rating** genannt. Ein Type-Rating ist somit die fachliche Umqualifizierung eines konkreten Piloten zu einem bestimmten Zeitpunkt.

Der Begriff **Schulung** wird im Sprachgebrauch sowohl im Sinne von *Kurs* als auch im Sinne von *Type-Rating* verwendet. Der Unterschied liegt in der nachfolgenden Präposition: Mit einer "Schulung *auf* PU q" ist ein Kurs gemeint, während eine "Schulung *von* PU p *nach* PU q" ein Type-Rating bezeichnet. Sprechen wir in Zukunft von einer *Schulung*, so wird entweder aus dem präpositionalen Zusammenhang klar werden, ob es sich um einen Kurs oder ein Type-Rating handelt, oder die Aussage wird für beide Möglichkeiten gleichermaßen zutreffen.

Ein **Schulungsplan** (**Kursplan**) ist eine Zusammenfassung von fixierten, virtuellen oder realen Kursen innerhalb eines gegebenen **Strategiezeitraumes**. Der Strategiezeitraum beginnt meist in der Gegenwart und endet 3 bis 5 Jahre später.

[1] Nachwuchsflugzeugführer

1.2.1 Ablauf der manuellen Schulungsplanung

Die manuelle Erstellung eines Schulungsplanes bei Lufthansa verläuft folgendermaßen:

(1) Bewerbung

Die Piloten reichen ihre Bewerbungen für die gewünschten Wechsel ein. Diese müssen zulässige Übergänge darstellen, d.h. jeweils auf eine PU schulen, die von der aktuellen PU aus als Ziel-PU erlaubt ist. Die Motivation für einen Wechsel sind einerseits deutlich höhere Gehälter (erstes FO-Jahresgehalt ca. 56 000 €, erstes Kapitänsgehalt ca. 110 000 €)[2], andererseits der Wechsel von der Kurz- auf die attraktivere Langstrecke. Zusätzlich können Bewerbungen für PUs eingereicht werden, auf die ein Wechsel erst nach dem Durchlaufen einer oder mehrerer Zwischen-PUs möglich wird. Da dies aber später keine bevorzugte Behandlung zur Folge hat, können wir annehmen, dass Bewerbungen tatsächlich immer nur entsprechend den zulässigen Übergängen eingereicht werden.

(2) Langfristige Schulungsplanung (SPTS)

Nach verschiedenen Kriterien wie Flugprogrammbedarf, Flottenauf-/abrüstungen und tariflichen Vereinbarungen werden vom menschlichen Planer fixierte Kurse innerhalb des Strategiezeitraumes angelegt. Dazu entscheidet er zum einen, wie viele Kurse überhaupt angelegt werden sollen und setzt zum anderen für jeden Kurs die unter 1.2 genannten Komponenten Starttermin, Ziel-PU und Anzahl der freien Plätze fest.

(3) Prognose

Für die angelegten fixierten Kurse ist zwar bekannt, wie viele Trainees auf die jeweiligen *Ziel-PUs* umschulen; für die Ressourcenplanung benötigen wir jedoch eine Hochrechnung der zukünftigen Pilotenbestände auf *allen* PUs und müssen deshalb auch vorhersagen, von welchen PUs die Piloten der fixierten Kurse stammen. Deshalb werden parallel zum Anlegen der fixierten Kurse in (2) vom manuellen Planer die Stamm-PUs der Trainees prognostiziert, also virtuelle Trainees eingeteilt. Diese Vorgehensweise ist neben den genannten Hochrechnungszwecken der Tatsache geschuldet, dass das bestehende Planungstool COMPAS[3] keine fixierten, sondern nur virtuelle Kurse verwaltet.

(4) Matching

In der Kurzfristplanung werden nun reale Piloten in die *fixierten* Kurse eingeteilt. Dabei spielt die in (3) gemachte Prognose keine Rolle mehr: Die Stamm-PUs hängen nun von den eingeteilten realen Piloten statt von prognostizierten virtuellen Trainees ab. Da nur im Ausnahmefall eine 1-1-Korrespondenz zwischen Bewerbern und freien Plätzen besteht, muss bei der Kurseinteilung eine Auswahl unter den Bewerbern getroffen werden (ein Mangel an Bewerbern bestehe aber nicht). Dies geschieht mittels der Seniorität. Die **Seniorität** ist eine jedem Piloten eindeutig zugeordnete Zahl, die seine Priorität bei Kurseinteilungen angibt. NFFs haben die schlechtesten Senioritäten, mit steigenden Dienstjahren oder verschiedenen Subfunktionen verbessert sich auch die Seniorität eines Piloten. Für die fixierten Kurse werden schließlich einfach diejenigen Bewerber mit den besten Senioritäten eingeteilt[4]. Dieser Vorgang heißt **Matching** und wandelt fixierte in reale Kurse um.

1.2.2 Ziele der automatischen Schulungsplanung

Die gute Wahl der fixierten Kurse in (2) ist das wesentliche Mittel und Problem der langfristigen Schulungsplanung. Mit den augenblicklichen Mitteln dauert die manuelle Erstellung eines Planes in der Regel 2 Tage oder länger. Ziel dieser Arbeit ist es, eine Automatisierung für die Erstellung guter Schulungspläne zu entwickeln. Daraus ergibt sich folgender Nutzen:

[2] Lufthansa Cockpit Careers: Cockpit, Karriereflugplan, Item 5, http://www.lufthansa-pilot.de (2004-07-01)

[3] Crew Operation Manpower Planning Advanced System

[4] Eine *gute* Seniorität wird intern durch eine kleine natürliche Zahl repräsentiert, eine *schlechte* durch eine große. Die Senioritäten können wir damit als Rangfolge interpretieren.

- Durch eine automatische Schulungsplanung wird die Berechnung von Szenarien beschleunigt. Dies ermöglicht in gleicher Zeit wie bisher die Bewertung von nun mehreren verschiedenen Szenarien und eine höhere Reaktionsgeschwindigkeit bei Veränderungen der Eingangsdaten.
- Eine automatische Schulungsplanung befähigt auch wenig erfahrene Planer dazu, Schulungspläne von guter Qualität zu erstellen.
- Bisher konnte die Güte von Schulungsplänen nur grob anhand von Kenngrößen eingeschätzt werden. Die Automatisierung komplexer Arbeitsgänge der Kursplanerstellung ermöglicht nun auch eine fundierte quantitative Bewertung.
- Durch bessere Planungsqualität lassen sich Schulungen einsparen.
- Durch schnellere und genauere Abstimmung der Schulungsplanung mit dem Netzbereich[5] kann besser abgeschätzt werden, ab welchen Zeitpunkten neu anzuschaffende Flugzeuge mit genügend Personal besetzt (**bereedert**) werden können.

Auf die in (3) angesprochene prognostische Umwandlung von fixierten zu virtuellen Kursen werden wir in Kapitel 5 zurückkommen. Das unter (4) beschriebene Matching ist im genannten System COMPAS bereits vollständig realisiert.

1.3 Gütekriterien

Unser Ziel in diesem Abschnitt wird es sein, eine genauere Vorstellung davon zu bekommen, was ein *guter Schulungsplan* ist. Der wesentliche Aspekt hierbei wird die Einführung des *Deltas*, der Differenz zwischen Bestand und Bedarf, in Abschnitt 1.3.3 sein. Alle Größen, die in Zusammenhang mit der Berechnung des Deltas auftauchen, beinhalten eine der beiden folgenden Einheiten: Eine **Einsatzstunde (eh)** ist eine Stunde, in der ein Pilot fliegerischen oder nichtfliegerischen Dienst leistet. Ein Tag, an dem er Einsatzstunden erbringt – egal wie viele – heißt **Einsatztag (et)**.

1.3.1 Bestand

Der **eh-Bestand** ist die auf einer PU an einem Tag verfügbare Menge an Einsatzstunden. Analog ist der **et-Bestand** die verfügbare Menge an Einsatztagen. Beide Größen werden mittels unterschiedlicher Prämissen berechnet und sind nicht direkt ineinander umwandelbar. Die Bestände setzen sich aus den folgenden nichtnegativen Summanden zusammen.

Bestand durch aktive Piloten

Der Großteil des eh-/et-Bestandes resultiert natürlich aus den Pilotenressourcen. Wir ziehen vom **Pilotenbestand** einer PU diejenigen Piloten ab, die in Mutterschutz, Erziehungsurlaub oder unbezahltem Sonderurlaub sind (prozentuale Abzüge in Abhängigkeit vom Pilotenbestand) oder lediglich in Teilzeit arbeiten (absolute Abzüge, da die Teilzeitkapazität absolut gegeben ist) und addieren ausgeliehene Piloten anderer Fluggesellschaften (**Ausgeliehene**) bzw. ziehen die Piloten ab, die an andere Airlines ausgeliehen sind (**Abgeordnete**) und erhalten den **aktiven Pilotenbestand**. Jeder aktive Pilot erbringt nun pro Tag eine gewisse Anzahl an Einsatzstunden, seine **eh-Produktivität**. Analog ist die **et-Produktivität** die Anzahl an Einsatztagen, die ein Pilot einer gewissen PU täglich erbringt. Die Produktivitätszahlen beschreiben Urlaubs- und Krankheitsabwesenheiten und ergeben, multipliziert mit der Anzahl aktiver Piloten, den durch aktive Piloten verfügbaren eh-/et-Bestand einer PU.

Betrachtet man einen konkreten Tag, so kann ein Pilot an diesem Tag entweder Dienst leisten oder nicht, d.h. seine Einsatztage-Produktivität an diesem Tag ist streng genommen entweder 0 oder 1. Bilden wir aber das arithmetische Mittel dieser Werte über einen bestimmten Zeitraum, so entstehen gebrochene et-Produktivitäten (bei eh analog). Besteht auf einer PU beispielsweise eine et-Produktivität von 0.5, dann erbringt ein Pilot dieser PU pro Monat etwa 15 Einsatztage.

[5] Der Netzbereich ist verantwortlich für die Erstellung des Flugplanes unter wirtschaftlichen Gesichtspunkten und bestmöglichem Kapazitätseinsatz.

Ein direkter Zusammenhang zwischen den beiden Produktivitätszahlen ist nicht gegeben: Es können auf zwei PUs p und q jeweils 80 Einsatzstunden pro Pilot und Monat erbracht werden, während diese auf PU p in 15 Einsatztagen und auf PU q in nur 12 Einsatztagen geleistet werden. Unterschiede in der Gewichtung treten zum Beispiel zwischen Lang- und Kurzstreckenflotten auf.

FO-Leistung

Der dritte Teil eines Kurses ist die Linienausbildung. Da aus Kostengründen keine Flugzeuge zu reinen Trainingszwecken verwendet werden können, findet die Linienausbildung vollständig auf regulären Flügen statt. Der Ausbilder ist immer ein speziell geschulter CP der Ziel-Flotte, ein sogenannter **Checker**. Die Linienausbildung nimmt er während seiner regulären Flugzeit vor:

- Ist ein FO auszubilden, sitzt der Checker wie im normalen Flugbetrieb auf dem CP-Stuhl im Cockpit und erbringt CP-Leistung, während der auszubildende FO auf dem FO-Stuhl sitzt und zusätzliche FO-Leistung bringt, denn der eigentliche FO dieses Fluges kann während dieser Zeit anderweitig eingesetzt werden.
- Ist ein CP auszubilden, sitzt *er* auf dem CP-Stuhl und erbringt die CP-Leistung des Checkers, während der Checker den FO-Stuhl einnimmt und damit zusätzliche FO-Leistung erbringt, da der eigentliche FO des Fluges wiederum anderweitig eingesetzt werden kann.

In beiden Fällen wird während der Linienausbildung zusätzliche **FO-Leistung** in Form von Einsatzstunden/-tagen erbracht, die den Bestand der entsprechenden FO-PUs erhöht.

Mehrflugstunden

Der eh-Bestand kann zusätzlich durch **Mehrflugstunden** (**Überstunden**) aufgestockt werden. Mehrflugstunden sind mit einem Lohnaufschlag für die Piloten verbunden und in ihrer Anzahl durch gesetzliche und tarifliche Regelungen beschränkt.

1.3.2 Bedarf

Dem Bestand gegenüber steht der Bedarf. Auch er wird in **eh-Bedarf** und **et-Bedarf** unterteilt und setzt sich aus den folgenden nichtnegativen Summanden zusammen.

Flugprogrammbedarf

Der **Flugprogrammbedarf** ist der durch den Flugplan fest vorgegebene tägliche Bedarf an Einsatzstunden/-tagen einer PU. Dazu zählen auch **Stand Bys** und **Stabilitätsreserven**; dies sind Bereitschaftspiloten, die bei Krankheit und Notfällen innerhalb von 1h bzw. 13h einsatzbereit sein müssen. Der Flugprogrammbedarf stellt im Normalfall den größten Teil des Bedarfes dar.

Sonstiger Bedarf

Unter dem **sonstigen Bedarf** wird der nichtfliegerische Bedarf verstanden, der pro Pilot entsteht. Er wird zum Beispiel durch folgende Aktivitäten verursacht:

- Lizenzerhaltende Maßnahmen wie Base Checks/Refresher (regelmäßige Nachschulungen), Emergency (Notfalltraining) und medizinische Untersuchungen
- Funktionärstätigkeit (Büroarbeiten) und sonstige Bodentätigkeiten
- Flexibilitätsreserve (Reserve für kurzfristige Flugplanänderungen)
- nur eh-Bedarf: Abwesenheitsstunden (Erholungsurlaub, Alterssonderurlaub, Bildungsurlaub, bezahlter Sonderurlaub, Krankheit, Kur)
- nur et-Bedarf: Seminare

Checker-Bedarf

Checker werden neben der Linienausbildung auch im zweiten Schulungsteil, dem Simulatortraining, benötigt. Die Einsatzstunden/-tage, die sie dafür aufbringen müssen, heißen **Checker-Bedarf**. Die bei der Linienausbildung benötigen Checker hingegen zählen nicht zum Checker-Bedarf, da die Linienausbildung während der regulären Dienstzeit der Checker stattfindet und damit bereits in den Flugprogrammbedarf eingeht.

Checker-Bedarf entsteht außerdem bei den regelmäßigen Nachschulungen der Piloten (Line-Checks/Simulator) und durch die Checker selbst aufgrund von Meetings, Seminaren und Reisen zu den Ausbildungsorten. Da Checker immer CPs sind, fällt sämtlicher Checker-Bedarf auf den entsprechenden CP-Planungseinheiten[6] an.

TraFO-Bedarf

Zur Durchführung des theoretischen Teils einer Schulung werden **TraFOs** (Training First Officers) benötigt. Die Einsatzstunden/-tage, die sie dafür aufbringen müssen, heißen **TraFO-Bedarf**. Da TraFOs immer FOs sind, fällt der TraFO-Bedarf stets auf den entsprechenden FO-PUs an.

1.3.3 Delta

Nachdem wir nun Bestand und Bedarf im Detail spezifiziert haben, können wir das **eh-Delta** einer PU an einem Tag als die Differenz zwischen eh-Bestand und eh-Bedarf definieren; das **et-Delta** wird analog definiert. Sprechen wir in Zukunft schlicht vom *Delta*, so gilt die getroffene Aussage sowohl für das eh-Delta als auch für das et-Delta.

Betrachten wir das Delta näher. Wird es negativ, so überwiegt der Bedarf gegenüber dem Bestand und wir haben ein Defizit an Einsatzstunden/-tagen. Dies verursacht zunächst Kosten durch Mehrflugstunden, bei größerer Unterdeckung möglicherweise sogar das Stehenlassen (**Grounden**) von Flugzeugen aus Personalmangel. Wird das Delta positiv, so überwiegt der Bestand gegenüber dem Bedarf und wir haben einen Überhang an Piloten. Die Folge ist, dass die Personalkapazitäten nicht entsprechend ihren Möglichkeiten ausgeschöpft werden und unnötige Pilotengehälter zu zahlen sind. Wirtschaftlich sinnvoll ist also offenbar ein Delta in der Nähe von 0, bei dem weder Kosten durch Über- noch durch Unterdeckung entstehen.

Der Grund, warum die gesamte Deltarechnung sowohl für Einsatztage als auch für Einsatzstunden durchgeführt wird, liegt darin, dass eh-Delta und et-Delta zwei verschiedene Kenngrößen für die Güte eines Schulungsplanes sind, die nicht direkt ineinander umgerechnet werden können. Es kann durchaus geschehen, dass auf einer Planungseinheit an einem Tag gleichzeitig eine Einsatztage-Überdeckung und eine Einsatzstunden-Unterdeckung besteht. Ein *Delta in der Nähe von* 0 bedeutet hier also, dass sowohl eh-Delta als auch et-Delta nahe bei 0 liegen sollen.

1.3.4 Schulungskosten

Neben dieser qualitativen Anforderung an einen Schulungsplan fließen in die Bewertung der Güte auch monetäre Größen ein. Dies sind Kosten, die direkt durch Schulungen verursacht werden: Kosten für Ausbilder und Schulungsräume sowie Material- und Simulatorkosten. Darüber hinaus sollten die Teilnehmerzahlen von Schulungen möglichst gerade oder durch 4 teilbar sein, da viele Kursaktivitäten kostengünstig mit je 2 oder je 4 Trainees durchgeführt werden können. Bleiben beispielsweise bei einer auf 4 Trainees ausgelegten Simulatoreinheit 1-3 Simulatorplätze frei, wird trotzdem ein Checker benötigt und der Strom einer 4-Personen-Simulatoreinheit verbraucht, kurz: die Ressourcen werden nicht optimal ausgenutzt und es entstehen unnötige Kosten.

Damit sind wir der Antwort auf die anfängliche Frage, was ein guter Schulungsplan ist, einen wichtigen Schritt näher gekommen: Ein guter Schulungsplan sollte zum einen durch Kurse den Pilotenbestand der PUs so steuern, dass das Delta auf allen PUs an allen Tagen möglichst nahe bei 0 liegt, und zum anderen die Gesamtkosten der angelegten Kurse möglichst klein halten.

[6] Eine *einer PU entsprechende CP-PU* bezeichnet die CP-PU, die der gleichen Flotte angehört. Eine *einem Kurs entsprechende CP-PU* bezeichnet die der Ziel-PU entsprechende CP-PU. Analoges gilt für FOs.

1.4 Nebenbedingungen

Beim Erstellen eines Schulungsplanes gibt es einige Startdaten, die von vornherein fest vorgegeben sind (Gleichungs-Nebenbedingungen). Außerdem sind gewisse Rahmenbedingungen einzuhalten, deren Verletzung zur Folge hätte, dass der Schulungsplan nicht oder nur unter inakzeptabel hohen Kosten durchführbar ist (Ungleichungs-Nebenbedingungen). Beide Arten von Nebenbedingungen sind auf jeden Fall einzuhalten, ansonsten ist der Schulungsplan unzulässig.

1.4.1 Gleichungs-Nebenbedingungen

Anfangspilotenbestand

Der Pilotenbestand der einzelnen PUs am ersten Tag des Strategiezeitraumes ist fest vorgegeben.

Absolute Zu-/Abgänge

Es soll möglich sein, zu bestimmten Zeitpunkten Piloten explizit von PUs zu entfernen oder hinzuzufügen. Dies sind zum Beispiel Quereinsteiger anderer Fluggesellschaften (**Ready Entries**).

Prozentuale Abgänge

Während des Strategiezeitraumes gehen von jeder PU eine gewisse Anzahl Piloten weg, bedingt durch Fluguntauglichkeit, Kündigungen, Altersabgänge oder Sterbefälle. Diese Anzahl ist von der Pilotenzahl abhängig und damit im Vorhinein noch nicht absolut bekannt, da sich die Pilotenzahl einer PU innerhalb des Strategiezeitraumes abhängig von den angelegten Kursen ändert.

Eingefrorene Kurse

Ein **eingefrorener Kurs** ist ein virtueller Kurs, der in einem Schulungsplan auf jeden Fall enthalten sein muss. Zum Beispiel können durch tarifliche Vereinbarungen gewisse Kurse fest vorgegeben sein, unabhängig von ihrem möglicherweise negativen Einfluss auf die Güte eines Schulungsplanes. Eingefrorene Kurse dürfen die unten stehende Nebenbedingung des Kursabstandes verletzen.

1.4.2 Ungleichungs-Nebenbedingungen

Traineebeschränkung

Da Trainingsressourcen wie Ausbilder und Simulatoren nur begrenzt zur Verfügung stehen, gibt es für Schulungen maximale Teilnehmerzahlen. Diese variieren je nach Ziel-PU.

Simulatorkapazitäten

Werden dieselben Simulatoren zeitgleich von mehreren Kursen genutzt, zum Beispiel weil diese auf dieselbe Flotte schulen, dürfen die Simulatorkapazitäten dabei nicht überschritten werden.

Kursabstände

Aus organisatorischen Gründen und wegen der Begrenztheit verschiedener Kursressourcen ist es wünschenswert, einen Mindestabstand für die Starttage von Kursen gleicher Ziel-PU vorzugeben.

Überstundengrenze

Die Anzahl der Mehrflugstunden ist durch tarifliche und gesetzliche Bestimmungen begrenzt. Aus Flexibilitätsgründen sollten außerdem Überstunden in der Langfristplanung vermieden werden.

Kapitel 2

Modellierung

Nachdem wir die Problemstellung SPTS verbal formuliert haben, werden wir sie im folgenden Kapitel in eine mathematische Form bringen. Dazu werden wir zunächst das verwendete abstrakte mathematische Modell definieren und danach SPTS in diese Form überführen.

2.1 Problemklasse

Ein **gemischt ganzzahliges Problem** (Mixed Integer Program) sei gegeben durch

$$\text{(MIP)} \qquad z = \min\{\, cx \,:\, x \in \mathbb{R}^n \times \mathbb{Z}^m,\ Ax \leq b \,\},$$

wobei A eine gegebene reellwertige $k\times(n+m)$-Matrix, b ein reellwertiger k-dimensionaler und c ein reellwertiger $(n+m)$-dimensionaler Spaltenvektor ist und k, n, m natürliche Zahlen sind. Die ersten n Komponenten des Vektors x heißen die **reellen Variablen**, die restlichen m Komponenten die **ganzzahligen Variablen** des MIP. Durch die Ungleichung $Ax \leq b$ sind k **Nebenbedingungen** an den Vektor x gegeben. Einen Vektor $x \in \mathbb{R}^n \times \mathbb{Z}^m$, der alle Nebenbedingungen erfüllt, nennen wir eine **zulässige Lösung** des MIP. Die Menge aller zulässigen Lösungen bezeichnen wir mit X. Um die zulässigen Lösungen hinsichtlich ihrer Güte zu vergleichen, wird jeder zulässigen Lösung durch die **Zielfunktion** $x \mapsto cx$ ein Wert zugewiesen. Eine zulässige Lösung x^* heiße **optimale Lösung**, wenn sie die Zielfunktion minimiert, d.h. wenn $\forall\, x \in X \;\; cx^* \leq cx$ gilt. In diesem Falle nennen wir $z = cx^*$ den **optimalen Zielfunktionswert** des gemischt ganzzahligen Problems.

Im Verlaufe dieses Kapitels werden wir SPTS schrittweise als MIP modellieren. Der Grund für diese Entscheidung ist, dass viele in der Problemstellung aufgeführten Nebenbedingungen durch lineare Ungleichungen formuliert werden können. Der Nachteil der Wahl eines MIP ist, dass wir *sämtliche* Nebenbedingungen, die wir formulieren wollen, in lineare Form bringen müssen, selbst wenn wir die Problemstellung dadurch an einigen Stellen nicht ideal abbilden können. Ebenso sind wir nun daran gebunden, unsere Zielformulierung *"Delta auf allen PUs an allen Tagen möglichst nahe bei 0 bei möglichst geringen Gesamtkosten"* mittels einer linearen Zielfunktion darzustellen. Wir werden während des folgenden Kapitels auf kritische Stellen bei der Modellierung hinweisen und eventuell Verbesserungsvorschläge angeben oder alternative Modelle benennen.

Wegen der umfangreichen einfließenden Datenmenge werden wir den Variablenvektor x, die Nebenbedingungsmatrix A, den rechte-Seite-Vektor b und den Zielfunktionsvektor c nicht getrennt voneinander einführen, sondern uns an der Struktur des Kapitels Problemstellung orientieren und im jeweils zu modellierenden Kontext immer nur die dort benötigten Variablen, Nebenbedingungen und Zielfunktionskoeffizienten definieren. Zur Strukturierung verwenden wir folgende Schreibweise:

D Bei diesem Symbol werden **Daten** für das MIP bereitgestellt. Von diesen Eingangswerten hängen sämtliche Nebenbedingungen und Zielfunktionskoeffizienten ab. Jede feste Menge von Eingangsdaten erzeugt eine zugehörige **Probleminstanz** von SPTS.

V Bei diesem Symbol wird eine neue **Variablenklasse** eingeführt. Der Übersichtlichkeit halber verwalten wir Variablen nicht einfach als reelle und ganzzahlige Komponenten des Vektors x, sondern teilen sie zusätzlich in Variablenklassen ein, die jeweils ein eigenständiges Konzept der Modellierung repräsentieren. Jede Variablenklasse wird durch einen hochgestellten Index gekennzeichnet ($x^{\mathrm{P}}, x^{\mathrm{C}}, \dots$). Die Variablen selbst tragen dann diese Bezeichnung und werden zur Unterscheidung mit tiefgestellten problemspezifischen Indizees versehen ($x^{\mathrm{P}}_{p,d}, x^{\mathrm{C}}_{q,e}, \dots$). Jede zulässige Lösung $x \in X$ von SPTS nennen wir **Schulungsplan**.

NB Bei diesem Symbol werden wir eine neue Klasse von **Nebenbedingungen** definieren. Genau wie bei den Variablenklassen verkörpert eine Nebenbedingungsklasse ein eigenes Konzept. Das MIP schließlich enthält mehrere Nebenbedingungen dieses Typs. Das Nebenbedingungskonzept der *Traineebeschränkung bei Kursen* impliziert beispielsweise im MIP genau eine Nebenbedingung pro Kurs.

Obwohl alle Nebenbedingungen laut Definition als $\leq$-Ungleichungen vorliegen sollen, können wir für unser MIP auch $\geq$-Ungleichungen und Gleichungen definieren, da sie sich problemlos zu $\leq$-Ungleichungen umformen lassen: $(a \in \mathbb{R}^{n+m}, r \in \mathbb{R})$

$$\begin{aligned} ax \geq r \quad &\iff \quad -ax \leq -r \\ ax = r \quad &\iff \quad -ax \leq -r \quad \wedge \quad ax \leq r \end{aligned}$$

ZF Bei diesem Symbol definieren wir die **Zielfunktionskoeffizienten** einer Variablenklasse. Alle Variablen, für die wir dies nicht tun, sollen automatisch Zielfunktionskoeffizient 0 haben.

Darüber hinaus werden wir die Eingangsdaten und Variablen mit physikalischen bzw. problemspezifischen Einheiten [in eckigen Klammern] versehen. Dies hat keinerlei Einfluss auf die Struktur des MIP, hilft aber beim Verständnis der Eingangsdaten und der Nebenbedingungen. Die benutzten problemspezifischen Einheiten sind [p] für Pilot, [eh] für Einsatzstunde und [et] für Einsatztag. Physikalisch gesehen ist eine Einsatzstunde [eh] genau dasselbe wie eine Stunde [h], das heißt, eine eh-Produktivität von 6 Einsatzstunden pro Tag entspricht 6 [eh/d] = 6[h]/24[h] = 1/4 [ ], allerdings lässt sich die Schreibweise als Quotient [eh/d] bequemer lesen. Dimensionslose Größen kennzeichnen wir durch [ ]. Ein Verzeichnis aller verwendeten Einheiten ist in Anhang B zu finden.

2.2 Basisdefinitionen

2.2.1 Planungseinheiten

Gegeben seien

- eine endliche Menge al von Fluggesellschaften (Airlines)
- eine endliche Menge ac von Flugzeug-Mustern (Aircraft)
- eine endliche Menge fc $\supseteq \{CP, FO\}$ von Cockpitfunktionen

und eine Menge $PU' \subseteq \mathrm{al} \times \mathrm{ac} \times \mathrm{fc}$ von Planungseinheiten mit

$$(u, v, w) \in PU' \quad \Longrightarrow \quad (u, v, CP), (u, v, FO) \in PU'. \tag{2.1}$$

Im Weiteren werden wir PU′s (u, v, w) mit u-v-w bezeichnen. Die durch u-v-$w \sim \hat{u}$-$\hat{v}$-$\hat{w} :\Longleftrightarrow u = \hat{u} \wedge v = \hat{v}$ definierte Äquivalenzrelation zerlegt PU' in eine Menge F' von Flotten. Damit besagt (2.1), dass jede Flotte wenigstens eine CP- und eine FO-Planungseinheit enthalten soll.

Um die jeweils verfügbaren NFFs im Modell zu verwalten, führen wir eine **zusätzliche Planungseinheit** *NFF* ein und setzen $PU = PU' \cup \{NFF\}$. Diese zusätzliche Planungseinheit existiert nur im Modell und gehört keiner realen Flotte an. Damit wir eine einheitliche Schreibweise verwenden können, fassen wir $\{NFF\}$ als eigene Flotte auf und setzen $F = F' \cup \{\{NFF\}\}$. Für $p \in PU'$ bezeichne fc(p) die Cockpitfunktion; für die NFF-PU setzen wir fc(NFF) := NFF. Sprechen wir in Zukunft von PUs und Flotten, werden wir in die Aussagen stets die NFF-PU mit einschließen.

2.2.2 Zeitdiskretisierung

Eine Teilmenge $I \subseteq \mathbb{Z}$ heiße **Intervall**, wenn $\exists\, t, t' \in \mathbb{Z} :\ I = [t, t'] \cap \mathbb{Z}$ gilt. Da Intervalle offenbar endliche Mengen sind, können wir $\underline{I} := \min I$ und $\overline{I} := \max I$ definieren. Den Strategiezeitraum identifizieren wir mit einem Intervall T der ganzen Zahlen. Jedes Element von T entspricht dann genau einem Tag des Strategiezeitraumes.

Die Starttermine von Schulungen werden in der Kurzfristplanung aus organisatorischen Gründen oftmals noch einige Tage vor- oder zurückverschoben. Daher ist es nicht sinnvoll, in der Langfristplanung bereits tagesgenaue Kurse anzulegen. Wir werden deshalb jeweils mehrere aufeinander folgende Tage zu einem Intervall zusammenfassen und einen Kursstart nur *in dieses Intervall* statt auf einen konkreten Tag des Intervalls legen. Eine Partition des Strategiezeitraumes in Intervalle heiße **Zeitdiskretisierung**. Es soll die Möglichkeit bestehen, für jede Planungseinheit eine eigene Zeitdiskretisierung zu definieren, je nachdem, wie tagesgenau auf dieser PU gerechnet werden soll. Es sei also D eine Funktion, die jeder Planungseinheit $p \in PU$ eine Zeitdiskretisierung D_p zuweist. Dabei seien der erste und der letzte Tag des Strategiezeitraumes jeweils einelementige Intervalle, um auf allen PUs die gleichen Anfangs- und Endintervalle zu haben. Das einem Intervall $d \in D_p$ vorangehende Intervall der Diskretisierung auf PU p bezeichnen wir mit $\mathbf{pre}_p d$, das nachfolgende Intervall analog mit $\mathbf{next}_p d$. Entsprechend der Mengenkardinalität sei $|d|$ die Länge des Intervalls. Für die Abbildung eines Tages auf das zugehörige Intervall von D_p schreiben wir ebenfalls D_p.

■ $T = \{1,2,3,4,5,6,7,8,9\},\ p = \text{DLH-A320-CP},\ D_p = \big\{\{1\},\{2,3,4\},\{5,6\},\{7,8\},\{9\}\big\}$

$\Longrightarrow \underline{\{2,3,4\}} = 2,\ \overline{\{5,6\}} = 6,\ \text{pre}_p\{9\} = \{7,8\} = \text{next}_p\{5,6\},\ |\{7,8\}| = 2,\ D_p(4) = \{2,3,4\}$

Ein **Knoten** sei ein Element der Menge $K = \{\,(p,d)\ :\ p \in PU,\ d \in D_p\,\}$. Zur Veranschaulichung stellen wir uns ein zweidimensionales Schema vor, in dem in der einen Richtung die PUs, in der anderen Richtung für jede PU die Intervalle ihrer Zeitdiskretisierung abgetragen sind; die so erzeugten Objekte sind genau die Knoten (Abbildung 2.1). Für alle Berechnungen innerhalb eines Knotens (p,d) werden wir per Konvention stets die Eingangsdaten des Endtages $\overline{d}$ zugrundelegen. Die Knoten werden die Objekte sein, mit denen wir die Variablenklassen indizieren. Damit hängt die Existenz zulässiger Lösungen (=Schulungspläne) auch von der Wahl der Diskretisierungen ab.

2.3 Pilotenfluss

Die grundlegenden Einheiten in unserem Modell, das "Material", mit dem wir arbeiten, sind die Piloten. Jeder Pilot befindet sich zu jedem Zeitpunkt des Strategiezeitraumes an genau einem Knoten. Von dort aus gibt es drei Möglichkeiten, was mit ihm geschehen kann:

(a) Er setzt die Tätigkeit auf seiner PU fort, geht also einfach zum nächsten Knoten der PU über. Die NFFs werden in diese Aussage mit eingeschlossen, denn auch sie setzen die Tätigkeit auf ihrer PU fort: Sie warten darauf, für einen Kurs auf ein Einstiegsmuster eingeteilt zu werden.

(b) Er scheidet durch Fluguntauglichkeit, Kündigung, Altersabgang oder Sterbefall aus.

(c) Er hat eine Bewerbung für eine andere PU abgegeben und wird aufgrund seiner Seniorität für einen Kurs auf diese PU eingeteilt. Im Schema verlässt er seine PU, sobald er den Kurs beginnt, und kommt erst am Knoten nach dem letzten Schulungstag ins System zurück – und zwar bei der neuen PU, falls er den Kurs bestanden hat, ansonsten bei der alten.

Bei der Langfristplanung kann unmöglich mit konkreten Piloten gerechnet werden, da die Bewerbungsentscheidungen und die Senioritäten der einzelnen Piloten nicht vorhergesagt werden können. Wir beziehen obige Betrachtung der *Möglichkeiten eines Piloten* daher nicht auf einzelne Piloten, sondern auf die gesamte Pilotenmenge, die sich in einem Knoten befindet: Ein Teil von ihr bleibt auf der PU und geht zum nächsten Knoten über, ein Teil geht in Schulung und ein weiterer Teil verschwindet aus dem System. Wir haben es mit einer anonymen Menge von Piloten zu tun, die durch unser System *fließt*. In diesem Abschnitt werden wir uns nun damit beschäftigen, diesen Pilotenfluss im MIP zu modellieren.

2.3.1 Pilotenvariablen

Der Pilotenbestand an den Knoten ist keine feste Eingangsgröße, sondern wird durch die während der Optimierung angelegten Schulungen beeinflusst. Wir führen also Variablen ein:

V Für jeden Knoten $(p, d) \in K$ gebe die reelle Variable $x^{\mathrm{P}}_{p,d}$ den **Pilotenbestand** [p] auf PU p am letzten Tag $\overline{d}$ des Intervalls d an.

Warum definieren wir den Pilotenbestand nicht als ganzzahlige Variablenklasse? Der Grund ist, dass wir den durch Fluguntauglichkeit etc. bedingten Pilotenabgang durch eine prozentuale Größe modellieren werden, da wir noch nicht genau wissen, an welchen Tagen die tatsächlichen Abgänge der Piloten stattfinden werden. Pro Tag geht also einfach ein gewisser Anteil an Piloten weg, wodurch eine **fraktionale** (nichtganzzahlige) Pilotenzahl übrig bleibt. Wir können diese Zahl als Erwartungswert von Piloten interpretieren.

D Die Funktion stay : $PU \times T \to [0, 1]$ gebe für (p, t) den Anteil [ ] der Piloten an, die während des Tages t nicht von PU p aufgrund von 2.3(b) abgehen (**Verweilwahrscheinlichkeit**).

Die Anfangspilotenbestände und die absoluten Zu-/Abgänge seien wie folgt gegeben:

D Die Funktion inc : $PU \times T \to \mathbb{Z}$ gebe für (p, t) den **Pilotenzugang** (increase) [p] auf PU p am Tag t an.

Ist der Pilotenzugang positiv, kommen Piloten zur PU hinzu, bei negativen Werten werden Piloten abgezogen. Mittels des Pilotenzugangs zur NFF-PU können wir einerseits neu ausgebildete NFFs an den Tagen, an denen sie planmäßig zur Verfügung stehen, der NFF-PU zuführen und können andererseits der NFF-PU ab einem gewissen Zeitpunkt einen sehr großen Pilotenbestand geben und anhand der ab da tatsächlich auf die Einstiegsmuster umgeschulten NFFs die Anzahl der in Zukunft benötigten NFFs ablesen. Damit von einer PU nicht mehr Piloten als vorhanden abgezogen werden können, sorgen wir noch dafür, dass der Pilotenbestand nirgends negative Werte annimmt:

$$\text{NB} \quad \forall\,(p, d) \in K \;\; x^{\mathrm{P}}_{p,d} \geq 0 \tag{2.2}$$

Nun werden wir die einzelnen Pilotenvariablen durch Flussgleichungen miteinander verknüpfen: Wir erhalten den Pilotenbestand bei Knoten (p, d), indem wir die Piloten des vorangegangenen Intervalls $\mathrm{pre}_p d$, die aufgrund der Verweilwahrscheinlichkeit nicht abgegangen sind, übernehmen und die absoluten Zugänge des aktuellen Knotens addieren:

$$\forall\,(p, d) \in K \quad x^{\mathrm{P}}_{p,d} \;=\; [\,\text{falls } \exists\, \mathrm{pre}_p d\,]\; \mathrm{stay}(p, \overline{d}\,)^{\,|d|} \cdot x^{\mathrm{P}}_{p,\,\mathrm{pre}_p d} \;+\; \sum_{t \in d} \mathrm{inc}(p, t) \tag{2.3}$$

In dieser **statischen Flussgleichung** sind alle Variablenwerte durch die Eingangsdaten bereits eindeutig bestimmt, die Pilotenflüsse bewegen sich ausschließlich entlang der PUs (Abbildung 2.1). Damit haben wir die Forderungen (a) und (b) aus Abschnitt 2.3 im Pilotenfluss modelliert. Um die endgültige Flussgleichung zu erhalten, werden wir im folgenden Abschnitt einen Pilotenfluss *zwischen* den PUs durch Schulungsvariablen ermöglichen und damit auch (c) realisieren.

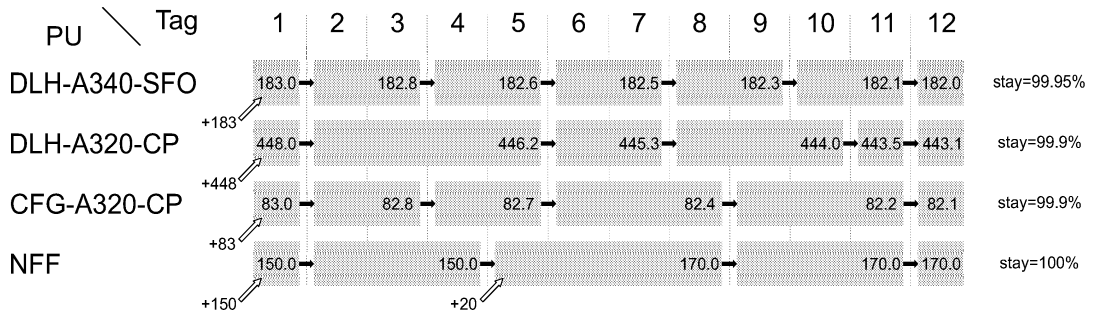


Abbildung 2.1: Beispiel eines statischen Pilotenflusses

2.3.2 Schulungsvariablen

V Für jeden Knoten $(p,d) \in K$ gebe die ganzzahlige Variable $x^{\mathrm{C}}_{p,d}$ die **Kursstärke** [p] des im Intervall d beginnenden fixierten Kurses auf (Ziel-)PU p an.

Wir haben damit jedem Knoten genau eine Schulungsvariable zugeordnet. Sprechen wir im Folgenden von der **Schulung** $x^{\mathrm{C}}_{p,d}$, meinen wir genauer die *dem Knoten* (p,d) *zugeordnete Schulungsvariable* $x^{\mathrm{C}}_{p,d}$. Nimmt diese den Wert 0 an, so startet am entsprechenden Knoten kein Kurs. Nimmt die Schulungsvariable einen positiven Wert an, so sagen wir, dass die Schulung $x^{\mathrm{C}}_{p,d}$ **angelegt** ist. Eine Schulung mit negativer Traineezahl ist natürlich nicht möglich:

NB $$\forall\,(p,d) \in K \quad x^{\mathrm{C}}_{p,d} \geq 0 \tag{2.4}$$

Durch die Traineebeschränkung existiert auch eine obere Schranke für die Schulungsvariablen:

D Die Funktion $\mathrm{cmax} : PU \times T \to \mathbb{N}_0$ gebe die **maximale Kursstärke** [p] eines Kurses an, der am Tag t auf PU p beginnt. Es gelte $\mathrm{cmax}(NFF, \cdot\,) = 0$.

NB $$\forall\,(p,d) \in K \quad x^{\mathrm{C}}_{p,d} \leq \mathrm{cmax}(p, \overline{d}\,) \tag{2.5}$$

Will der Anwender aus irgendeinem Grund verbieten, an einem Knoten (p,d) eine Schulung anzulegen, braucht er nur bei den Eingangsdaten $\mathrm{cmax}(p,\overline{d}\,) = 0$ zu setzen. Dies ist auf der gesamten NFF-PU der Fall: Sie wurde dazu eingeführt, um das Warten der NFFs auf Schulungen der Einstiegsmuster zu modellieren; es werden daher keine Schulungen *auf* die NFF-PU angelegt.

2.3.3 Stammverteilung

Die Schulungsvariablen repräsentieren fixierte Kurse. Es ist also für jeden Kurs lediglich die Ziel-PU bekannt; die Stamm-PUs der Trainees werden erst in der Kurzfristplanung durch das Matching mit realen Piloten festgelegt. Wollen wir allerdings den Pilotenfluss modellieren, müssen wir eine Möglichkeit finden, beim Anlegen einer Schulung $x^{\mathrm{C}}_{p,d}$ vorherzusagen, *woher* die Piloten stammen, die an dieser Schulung teilnehmen werden, damit wir sie im Modell von ihrer Stamm-PU auf die neue PU p fließen lassen können. Eine exakte Vorhersage der Stamm-PUs ist offensichtlich nicht möglich, da die konkreten Bewerbungsentscheidungen der realen Piloten in den nächsten Jahren nicht voraussehbar sind; wir können aber versuchen vorherzusagen, von welchen PUs die Piloten *wahrscheinlich* stammen.

D Die Funktion W: $PU \times PU \times T \to [0,1]$ heiße **Stammverteilung** [ ] und erfülle folgende beiden Eigenschaften:

$$(\mathrm{W}_1) \qquad \forall\, p \in PU \quad \forall\, t \in T \quad \mathrm{W}(p,p,t) = 0$$

$$(\mathrm{W}_2) \qquad \forall\, p \in PU \quad \forall\, t \in T \quad \sum_{q \in PU} \mathrm{W}(q,p,t) = 1$$

Der Wert $\mathrm{W}(q,p,t)$ gebe die Wahrscheinlichkeit dafür an, dass ein Trainee, der an einer an Tag t beginnenden Schulung auf PU p teilnimmt, von PU q stammt.

Die Stammverteilung wird aus historischen Schulungsdaten berechnet oder aus voraussehbaren Flottenentwicklungen und der Bewerbersituation prognostiziert. Eigenschaft (W_1) besagt, dass Trainees niemals an Kursen auf ihre eigene PU teilnehmen, denn darauf sind sie bereits geschult. **Rückschulungen** sind allerdings möglich; dies sind Wechsel auf bereits durchlaufene Planungseinheiten. Beispielsweise kann nach dem Erreichen der Langstreckenflotten aus familiären Gründen eine Rückschulung auf eine Kurzstreckenflotte erwünscht sein. Mit Eigenschaft (W_2) ist für jeden Kurs $x^{\mathrm{C}}_{p,d}$ eine Wahrscheinlichkeitsverteilung $\mathrm{W}(\,\cdot\,, p, \overline{d}\,)$ der Stamm-PUs gegeben.

Damit haben wir geklärt, woher die Trainees eines fixierten Kurses *wahrscheinlich* stammen: Wird an Tag t ein Kurs mit n Trainees auf PU p angelegt, so stammen durchschnittlich $\mathrm{W}(q,p,t)\cdot n$ Trainees von PU q. Im Flussmodell tun wir nun Folgendes: Wir ziehen von jeder Stamm-PU *tatsächlich* genau die laut Stammverteilung zu erwartende Pilotenmenge ab und lassen sie auf die Ziel-PU des Kurses fließen. Obwohl die Gesamtteilnehmerzahlen der Schulungen ganzzahlig sind, werden wir als Teilnehmer also keine *ganzen* Trainees, sondern nur *zu erwartende* Trainees fließen lassen. Diese nichtganzzahligen Trainees nennen wir **fraktionale Trainees**. Der Vorteil dieser Vorgehensweise ist, dass wir uns die Arbeit dadurch wesentlich vereinfachen: Wir müssen in der Optimierung lediglich fixierte Kurse anlegen (Starttag, Ziel-PU, Teilnehmerzahl); um die Prognose der Stamm-PUs und die Anzahl der Trainees, die von den verschiedenen Stamm-PUs kommen, müssen wir uns nicht mehr kümmern, sie sind durch die Stammverteilung deterministisch gegeben. Wollen wir in einer Nebenbedingung die Anzahl der fraktionalen Trainees von PU q verwenden, die an der Schulung $x^{\mathrm{C}}_{p,d}$ teilnehmen, benutzen wir folglich einfach den Term $\mathrm{W}(q,p,\overline{d}\,)\cdot x^{\mathrm{C}}_{p,d}$.

- Nehmen wir an, dass an Kursen auf eine bestimmte PU p nur Trainees von zwei verschiedenen Stamm-PUs q_1 und q_2 teilnehmen können. Aus den in unten stehender Tabelle dargestellten Zusammensetzungen von Kursen auf PU p, die bereits *vor* Beginn des Strategiezeitraums T stattgefunden haben, könnte man die Stammverteilung etwa wie folgt prognostizieren:

Starttag	$\underline{T}-205$	$\underline{T}-130$	$\underline{T}-67$
Trainees	4	8	8
Herkunft	1 von q_1, 3 von q_2	1 von q_1, 7 von q_2	2 von q_1, 6 von q_2

$$\mathrm{W}(q_1,p,\cdot) := \frac{1+1+2}{4+8+8} = \frac{1}{5} = 20\%$$
$$\mathrm{W}(q_2,p,\cdot) := \frac{3+7+6}{4+8+8} = \frac{4}{5} = 80\%$$

  Wird nun während der Optimierung ein Kurs mit 8 Trainees auf PU p angelegt, ziehen wir im Flussmodell $20\%\cdot 8 = 1,6$ Piloten von PU q_1 und $80\%\cdot 8 = 6,4$ Piloten von PU q_2 ab.

Der große Nachteil von fixierten Kurszusammensetzungen ist, dass die Anzahl der von den PUs kommenden Trainees nun allein von der Schulungsstärke abhängig ist und nicht von den verfügbaren Piloten auf den Stamm-PUs. Nehmen zum Beispiel laut Stammverteilung an Kursen auf eine gewisse PU p auch Trainees von PU q teil und hat PU q in einem bestimmten Zeitraum keine Piloten mehr, so können in diesem Zeitraum auch keine Kurse auf PU p angelegt werden, da diese auch Trainees von PU q abziehen würden, was aber nicht möglich ist. In unserem Modell müssen also zunächst Piloten auf PU q nachschulen, bevor wieder Kurse auf PU p angelegt werden können.

2.3.4 Flussgleichungen

Im vorhergehenden Abschnitt haben wir Variablen bereitgestellt, um einen Pilotenfluss zwischen den PUs zu ermöglichen. Ist eine Schulung $x^{\mathrm{C}}_{p,d}$ angelegt, werden Piloten entsprechend der Stammverteilung von anderen PUs abgezogen. Allerdings werden nicht alle Trainees am Starttag $\overline{d}$ der Schulung von ihrer PU abgezogen. Für einige beginnt die Schulung erst später; bis dahin dienen sie weiter auf ihrer alten PU. Da der offizielle Kursstart trotzdem schon bei Tag $\overline{d}$ liegt, bezeichnen wir die Zeit bis zum tatsächlichen Kursbeginn dieser Trainees als **Schulungsteil 0**. Der Endtag von Kursen ist aber für alle teilnehmenden Trainees derselbe. Die unterschiedlichen Schulungslängen resultieren aus den verschiedenen Vorkenntnissen der Piloten verschiedener PUs. Eine Umschulung von einer Airbus- auf eine Boeing-Flotte wird zum Beispiel länger dauern als eine Umschulung zwischen zwei Airbusmustern. Mit der Schulungslänge variiert für Trainees desselben Kurses auch die Länge der 3 Schulungsteile, je nachdem, von welcher PU die Trainees stammen. Schulungsdauern werden also nicht pro Ziel-PU gegeben, sondern pro Übergang zwischen zwei PUs.

D Die Funktionen $\mathrm{dur}_i : PU \times PU \times T \to \mathbb{N}_0$ geben für (q,p,t) die **Dauer** [d] des i-ten Schulungsteiles ($i = 1,2,3$) für Piloten von PU q an, die an einer an Tag t beginnenden Schulung auf PU p teilnehmen.

Kurse auf dieselbe PU sind in unserem Modell wegen der Einführung der Stammverteilung stets gleich zusammengesetzt: An einer Schulung nehmen fraktionale Trainees *aller* Stamm-PUs teil, gewichtet entsprechend der Stammverteilung. Die Gesamtdauer dur eines Kurses – vom offiziellen Starttag bis zum gemeinsamen Endtag aller Trainees – setzen wir daher auf das Maximum über die Schulungsdauern der einzelnen teilnehmenden fraktionalen Trainees. Damit können wir auch die Dauer dur_0 des Schulungsteils 0 bestimmen:

$$\begin{aligned} \mathrm{dur}(p,t) &:= \max_{\substack{q \in PU \\ \mathrm{W}(q,p,t)>0}} \sum_{i=1}^{3} \mathrm{dur}_i(q,p,t) \\ \mathrm{dur}_0(q,p,t) &:= \mathrm{dur}(p,t) - \sum_{i=1}^{3} \mathrm{dur}_i(q,p,t) \end{aligned}$$

Offenbar gibt es für jeden Kurs mindestens eine Stamm-PU, für die Schulungsteil 0 verschwindet, deren Trainees also tatsächlich ab dem offiziellen Kursstart in Schulung gehen, wodurch der Begriff *Kursstart* gerechtfertigt bleibt. Mit der Kenntnis der Schulungsdauern besitzen wir nun Daten darüber, wann Piloten, die in Schulung gehen, von ihren Stamm-PUs abgezogen werden und wann sie fertig geschult sind. Hierbei heiße ein Trainee genau am Tag nach Kursende **fertig geschult**.

$D^{\mathrm{in}}_{p,d}$ sei im Weiteren die Menge der Intervalle auf PU p, deren Kurse ihre Trainees im Intervall d fertig geschult haben:

$$\forall\,(p,d) \in K \quad D^{\mathrm{in}}_{p,d} := \left\{\, e \in D_p \,:\, \overline{e} + \mathrm{dur}(p,\overline{e}) \in d \,\right\}$$

$D^{\mathrm{out}}_{q,p,d}$ sei die Menge der Intervalle auf PU q, deren Kurse Piloten von PU p im Intervall d abziehen:

$$\forall\, q \in PU \quad \forall\,(p,d) \in K \quad D^{\mathrm{out}}_{q,p,d} := \left\{\, e \in D_q \,:\, \overline{e} + \mathrm{dur}_0(p,q,\overline{e}) \in d \,\right\}$$

Welcher Anteil der Trainees den Kurs nicht erfolgreich absolviert und deshalb wieder auf seine vorherige PU zurückkehrt bzw. welcher Anteil nach Schulungsende tatsächlich auf die Ziel-PU übergeht, modellieren wir durch folgende Eingangsgröße:

D Die Funktion $\mathrm{fail} : PU \times PU \times T \to [0,1]$ gebe für (q,p,t) die **Wahrscheinlichkeit** [ ] dafür an, dass ein Pilot von PU q eine an Tag t beginnende Schulung auf PU p nicht besteht.

Jetzt haben wir alle Daten gesammelt, um die statische Pilotenflussgleichung (2.3) zur endgültigen Flussgleichung zu ergänzen, die auch den auf Seite 10 unter Punkt (c) beschriebenen Pilotenfluss *zwischen* den PUs berücksichtigt. Sinn der Flussgleichungen ist es, die Pilotenbestände an den Knoten zu berechnen. Überlegen wir uns, welche Komponenten einfließen: Den Pilotenbestand bei einem Knoten (p,d) erhalten wir, indem wir die Piloten des vorangegangenen Intervalls, die nicht durch Fluguntauglichkeit etc. abgegangen sind, übernehmen und dazu die absoluten Zugänge des aktuellen Knotens addieren (statischer Pilotenfluss). Es sind nun diejenigen Piloten abzuziehen, die eine Umschulung auf eine andere PU beginnen, und analog diejenigen Trainees zu addieren, die gerade erfolgreich eine Schulung auf PU p abgeschlossen haben[1] oder eine Schulung auf eine andere PU nicht bestanden haben und deshalb wieder zu PU p zurückkehren (Abbildung 2.2). Damit sehen die endgültigen Flussgleichungen wie folgt aus:

[1] In unserem Modell kommen Trainees sofort nach erfolgreichem Kursende zum Pilotenbestand der Ziel-PU hinzu und haben damit sofort wieder die Möglichkeit, an einem Kurs auf eine weitere PU teilzunehmen. Wir haben die in der Problemstellung genannte *Mindestverweildauer nach Schulungen* aus zwei Gründen nicht modelliert: Erstens schulen wir im Modell nur anonyme Pilotenmengen um, d.h. wir können annehmen, dass die Teilnehmer eines Kurses nicht gerade diejenigen Piloten einer PU sind, die eben erst von einem Kurs angekommen sind. Zweitens ist die Mindestverweildauer ein Steuerungsmittel des Planers und kann von ihm gesenkt werden, wenn eine Notwendigkeit dazu besteht. Im Übrigen würde die Hinzunahme der Mindestverweildauer ins Modell keine Schwierigkeiten bereiten: Wir legen analog zu den Pilotenvariablen *Verweilvariablen* an, auf die Piloten nach erfolgreicher Schulung zunächst fließen. Jene Piloten zählen wir bei Rechnungen jeweils schon zu den Pilotenbeständen, aber erst nach Ablauf der Mindestverweildauer lassen wir sie auf die regulären Pilotenvariablen fließen, von wo aus sie wieder schulen können.

NB $\forall\,(p,d) \in K$

$$x^{\mathrm{P}}_{p,d} \;=\; [\,\text{falls } \exists\ \mathrm{pre}_p d\,]\ \mathrm{stay}(p,\overline{d}\,)^{\,|d|} \cdot x^{\mathrm{P}}_{p,\,\mathrm{pre}_p d} \;+\; \sum_{t \in d} \mathrm{inc}(p,t) \;-\; \sum_{q \in PU} \sum_{e \in D^{\mathrm{out}}_{q,p,d}} \mathrm{W}(p,q,\overline{e}\,) \cdot x^{\mathrm{C}}_{q,e}$$

$$+ \sum_{q \in PU} \sum_{e \in D^{\mathrm{in}}_{p,d}} \big(1 - \mathrm{fail}(q,p,\overline{e})\big) \cdot \mathrm{W}(q,p,\overline{e}\,) \cdot x^{\mathrm{C}}_{p,e} + \sum_{q \in PU} \sum_{e \in D^{\mathrm{in}}_{q,d}} \mathrm{fail}(p,q,\overline{e}) \cdot \mathrm{W}(p,q,\overline{e}\,) \cdot x^{\mathrm{C}}_{q,e} \qquad (2.6)$$

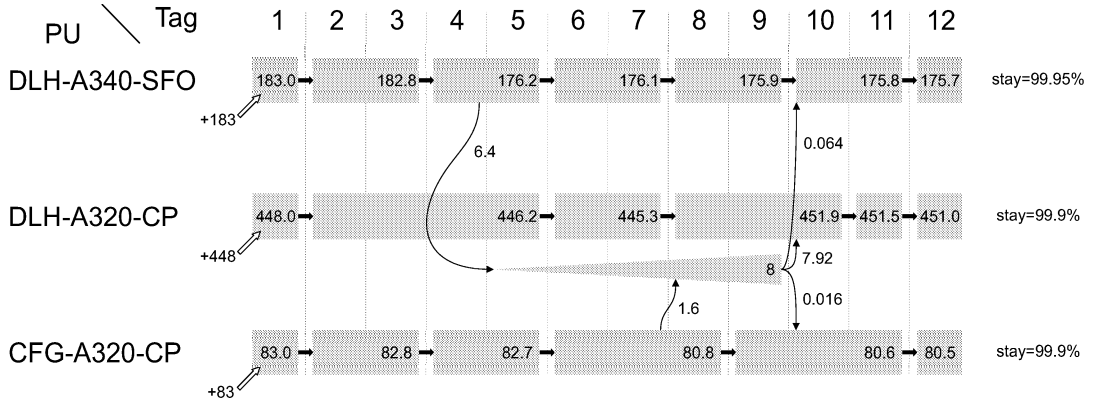


Abbildung 2.2: Beispiel eines Pilotenflusses. An Tag 5 beginnt ein Kurs auf DLH-A320-CP für 8 Trainees. Die Stammverteilung sei so gegeben, dass 80% der Trainees von DLH-A340-SFO und 20% von CFG-A320-CP kommen. Da letztgenannte Trainees lediglich die Fluggesellschaft wechseln, während Cockpitfunktion und Flugzeug-Typ gleich bleiben, dauert ihre Umschulung wesentlich kürzer. Bei einer angenommenen Durchfallrate von 1% bestehen erwartungsgemäß 7.92 Trainees; die anderen gehen zurück auf ihre Stamm-PUs. Zur Vereinfachung wurde die Schulungslänge im Beispiel auf wenige Tage beschränkt; in Wirklichkeit dauern Schulungen bis zu einem halben Jahr.

2.4 Kursbeschränkungen

2.4.1 Eingefrorene Kurse

Einige Kurse sind vom Planer bereits fest gegeben (eingefroren) und müssen in jedem zulässigen Schulungsplan enthalten sein. Es handelt sich hierbei um eine Menge virtueller Kurse: Die Stamm-PUs der Trainees sind schon vorgegeben. Während des Matchings bleiben diese Kurse unberührt; der menschliche Planer teilt später reale Trainees entsprechend der eingefrorenen Stamm-PUs ein. Als Eingangsdaten lassen wir uns die einzelnen Übergänge der eingefrorenen Kurse geben:

D Die Funktion freez : $PU \times PU \times T \to \mathbb{N}_0$ gebe für (q,p,t) die **Anzahl der Trainees** [p] von PU q an, die an dem eingefrorenen Kurs auf PU p teilnehmen, der an Tag t beginnt. Auf die NFF-PU existieren keine eingefrorenen Kurse, es gelte $\mathrm{freez}(\,\cdot\,, \mathit{NFF}, \cdot\,) = 0$.

Für eine feste Ziel-PU p und einen festen Tag t gibt die Funktion $\mathrm{freez}(\,\cdot\,,p,t)$ die Verteilung der Stamm-PUs des an Tag t startenden eingefrorenen Kurses auf PU p an. Ist diese Funktion konstant 0, so ist kein Übergang gegeben und bei (p,t) findet kein eingefrorener Kurs statt.

Für einen eingefrorenen Kurs bei (p,t) kennen wir bereits den genauen Starttag. Wir müssen uns also nicht auf ein Diskretisierungsintervall beschränken, *in dem* der Starttag des Kurses liegt, sondern wählen die Diskretisierung D_p so, dass das eintägige Intervall $\{t\}$ selbst enthalten ist:

$$\forall\, p \in PU \quad D^{\mathrm{freez}}_p := \big\{\{t\} :\ \exists\, q \in PU\ \exists\, t \in T\ \ \mathrm{freez}(q,p,t) > 0\big\} \ \subseteq\ D_p$$

Fixieren wir nun die Schulungsvariablen, die den eingefrorenen Kursen entsprechen, auf die Anzahl der teilnehmenden Trainees. Sie ist die Summe der einzelnen Übergänge des eingefrorenen Kurses:

$$\boxed{\text{NB}}\quad \forall p \in PU\ \ \forall t \in T\ \ \big(\text{mit } \exists q \in PU\ \ \text{freez}(q,p,t) > 0\big)\ \ x^{\text{C}}_{p,\{t\}} = \textstyle\sum_{q \in PU} \text{freez}(q,p,t) \tag{2.7}$$

Dass wir auch die eingefrorenen Kurse durch Schulungsvariablen darstellen, welche wir eigentlich zur Repräsentierung fixierter Kurse geschaffen hatten, hat zur Folge, dass wir momentan auch bei den eingefrorenen Kursen eine Stamm-PU-Prognose mittels der Stammverteilung vornehmen. Um die Trainees nun korrekt entsprechend der freez-Vorgabe statt der Stammverteilung abzuziehen, passen wir die Stammverteilung an den Tagen, an denen eingefrorene Kurse starten, an:

$$\forall p \in PU\ \ \forall t \in T\ \ \big(\text{mit } \exists q \in PU\ \ \text{freez}(q,p,t) > 0\big)\ \ \forall q \in PU\ \ \text{W}(q,p,t) := \frac{\text{freez}(q,p,t)}{x^{\text{C}}_{p,\{t\}}}$$

2.4.2 Teilbinäre Kursdarstellung

Da wir für jeden Knoten eine eigene Kursvariable definiert haben, kann im jetzigen Modell auch bei jedem Knoten eine Schulung angelegt werden bzw. ist schon eine eingefrorene Schulung angelegt. In Wirklichkeit existieren jedoch beschränkte Trainingsressourcen, die die Anzahl der in einem bestimmten Zeitraum durchführbaren Kurse begrenzen. Weil viele Ressourcen jeweils pro 2 oder 4 Trainees anfallen, müssen wir eine Möglichkeit finden, die Traineezahlen von Kursen für die Berechnung der von ihnen gebundenen Ressourcen auf das jeweils nächste Vielfache von 2 oder 4 aufzurunden. Die Schwierigkeit ist, dass dieses Aufrunden mit Mitteln eines MIP geschehen muss, d.h. mit linearen Gleichungen und Ungleichungen. Schlüssel zur Lösung ist die teilbinäre Darstellung der Teilnehmerzahlen. Die **teilbinäre Darstellung** zerlegt die Teilnehmerzahl n eines Kurses in die Anzahl n_4 der teilnehmenden Pilotenquadrupel und die (in n_2 und n_1 binär codierte) Anzahl der verbleibenden Piloten, die kein Quadrupel mehr bilden; es gilt also $n = 4n_4 + 2n_2 + n_1$. Offensichtlich ist jede Teilnehmerzahl eindeutig auf diese Weise darstellbar.

■

n	0	1	2	3	4	5	6	7	8	9	10	11
$n_4\, n_2 n_1$	0 00	0 01	0 10	0 11	1 00	1 01	1 10	1 11	2 00	2 01	2 10	2 11

Modellieren wir die teilbinäre Zerlegung der Kursvariablen $x^{\text{C}}_{p,d}$ nun im MIP:

$\boxed{\text{V}}$ Für $(p,d) \in K$ gebe die ganzzahlige Variable $x^{(4)}_{p,d}$ die **Anzahl der Pilotenquadrupel** [4p] an, die an Kurs $x^{\text{C}}_{p,d}$ teilnehmen. Die ganzzahligen Variablen $x^{(2)}_{p,d}$ und $x^{(1)}_{p,d}$ kennzeichnen die Teilnahme eines **zusätzlichen Pilotenpaares** [2p] bzw. eines **zusätzlichen Piloten** [p].

$$\boxed{\text{NB}}\quad \forall (p,d) \in K\ \ 0 \le x^{(2)}_{p,d} \le 1,\ 0 \le x^{(1)}_{p,d} \le 1 \tag{2.8}$$

Damit haben wir eine ganzzahlige Variablenklasse $x^{(4)}$ und zwei binäre Variablenklassen $x^{(2)}$ und $x^{(1)}$ definiert. Aufgrund der gewählten Definitionsbereiche dieser Variablen und der Eindeutigkeit der teilbinären Darstellung genügt folgende Nebenbedingung, um die Kurse teilbinär zu zerlegen:

$$\boxed{\text{NB}}\quad \forall (p,d) \in K\ \ x^{\text{C}}_{p,d} = 4x^{(4)}_{p,d} + 2x^{(2)}_{p,d} + x^{(1)}_{p,d}. \tag{2.9}$$

Finden nun gewisse Schulungsaktivitäten *pro Traineepaar* statt, so bindet ein einzelner Trainee die gleichen Ressourcen wie ein Paar; ein Quadrupel benötigt die gleichen Ressourcen wie zwei Paare. Damit fallen Schulungsressourcen für insgesamt $2x^{(4)} + x^{(2)} + x^{(1)}$ Traineepaare an. Finden Schulungsaktivitäten *pro Traineequadrupel* statt, bindet ein einzelner Trainee oder ein Paar jeweils die Ressourcen eines vollen Quadrupels. Wir berechnen die Schulungsressourcen in diesem Fall für insgesamt $x^{(4)} + x^{(2)} + x^{(1)}$ Traineequadrupel.[2]

[2] Diese Formel liefert im Fall von 3 zusätzlichen Trainees ($x^{(2)} = x^{(1)} = 1$) die Bindung der Ressourcen von *zwei* Quadrupeln, obwohl für 3 Trainees eigentlich die Ressourcen *eines* Quadrupels ausreichen. Da wir aber das Aufrunden auf Traineequadrupel später nur zum Berechnen von Strafkosten verwenden werden (dafür, dass keine reinen Quadrupel geschult werden, sondern Piloten übrigbleiben), ist eine Modellierung auf diese Weise ausreichend. Eine korrekte Abbildung ist über die Einführung einer weiteren Binärvariable möglich: Definieren wir für $y \in \{0,1\}$ die beiden Nebenbedingungen $x^{(2)} \le y$ und $x^{(1)} \le y$, beträgt die Anzahl der gebundenen Pilotenquadrupel $x^{(4)} + y$.

2.4.3 Simulatorkapazität

Während des zweiten Schulungsteiles müssen Trainees eine vorgeschriebene Menge an Simulatoreinheiten erbringen. Eine Simulatoreinheit ist immer für 2 oder für 4 Piloten ausgelegt, jeder Pilot muss eine gewisse Anzahl beider Typen absolvieren. Wir geben die pro 2 und pro 4 Trainees zu leistenden Simulatoreinheiten in Einsatztagen und – hier ist dies möglich – mittels eines direkten Umrechnungsfaktors auch in Einsatzstunden an.

D Die Funktion $\mathrm{sim}^{2p} : PU \times PU \times T \to [0,1]$ gebe bei (q,p,t) für eine an Tag t beginnende Schulung den **Simulator-Einsatztagebedarf** [et/2p d] pro Traineepaar, das von q nach p schult, und pro Tag des 2. Schulungsteils an.

D Die Funktion $\mathrm{sim}^{4p} : PU \times PU \times T \to [0,1]$ gebe bei (q,p,t) für eine an Tag t beginnende Schulung den **Simulator-Einsatztagebedarf** [et/4p d] pro Traineequadrupel, das von q nach p schult, und pro Tag des 2. Schulungsteils an.

D Die Funktion $\mathrm{sim}^{\mathrm{et}\to\mathrm{eh}} : PU \times T \to [0,24]$ gebe bei (p,t) für eine an Tag t beginnende Schulung auf PU p den im 2. Schulungsteil bestehenden **Simulator-Umrechnungsfaktor** [eh/et] von Einsatztagen zu Einsatzstunden an.

Für jede Flotte existiert nur eine begrenzte Zahl an Flugsimulatoren und damit eine tägliche Maximalkapazität an zur Verfügung stehenden Simulatorstunden.

D Die Funktion $\mathrm{sim} : F \times T \to \mathbb{R}_+$ gebe für (f,t) die an Tag t zur Verfügung stehenden **Simulator-Einsatzstunden** [eh] auf Flotte f an.

Stammen alle Trainees eines Kurses $x^{\mathrm{C}}_{p,d}$ von derselben PU q, dann beträgt der Simulatorstunden-Bedarf pro Tag des 2. Schulungsteils

$$\mathrm{sim}^{\mathrm{et}\to\mathrm{eh}}(p,\overline{d}\,)\cdot\Big(\mathrm{sim}^{4\mathrm{p}}(q,p,\overline{d}\,)\cdot\big(x^{(4)}_{p,d}+x^{(2)}_{p,d}+x^{(1)}_{p,d}\big)+\ \mathrm{sim}^{2\mathrm{p}}(q,p,\overline{d}\,)\cdot\big(2x^{(4)}_{p,d}+x^{(2)}_{p,d}+x^{(1)}_{p,d}\big)\Big).$$

Weil Trainees jedoch im Allgemeinen von verschiedenen PUs stammen und obige Traineequadrupel bzw. -paare deswegen gemischt sind, gewichten wir den Simulatorbedarf entsprechend der Stammverteilung und rechnen folglicherweise mit dem *zu erwartenden* Simulatorbedarf.

Die Einhaltung der Simulatorkapazität fordern wir *pro Flotte* und nicht *pro PU*, da alle PUs einer Flotte dieselben Simulatoren benutzen. Obwohl in der Realität die Simulatorkapazitäten *pro Tag* nicht überschritten werden dürfen, werden wir ihre Einhaltung jeweils nur *in Intervallen* prüfen. Dies vergrößert zwar die zulässige Lösungsmenge ein wenig, weil es zu Kapazitätsumverteilungen innerhalb der Intervalle kommen kann, wird jedoch dadurch ausgeglichen, dass die kurzfristige Simulatorplanung jene Umverteilung kompensieren kann. Als Überprüfungsintervalle können wir, da jeweils nur pro Flotte geprüft werden soll, die Diskretisierung einer PU der Flotte verwenden. Als Konvention wählen wir jeweils die Diskretisierungen der CP-PUs. Unsere Nebenbedingung pro Flotte und Intervall lautet in Worten: Der zu erwartende Simulatorbedarf, den alle Kurse auf alle PUs dieser Flotte, die sich während des Intervalls im 2. Schulungsteil befinden, erzeugen, darf die im Intervall zur Verfügung stehende Simulatorkapazität nicht überschreiten. Zur besseren Übersicht fassen wir jene Intervalle von PU p, in denen Kurse starten, deren Trainees von PU q am Tag t im 2. Schulungsteil sind, in der Menge

$$D^2_{q,p,t} := \Big\{\, d \in D_p \,:\, \overline{d} + \sum_{i=0}^{1} \mathrm{dur}_i(q,p,\overline{d}\,) \le t < \overline{d} + \sum_{i=0}^{2} \mathrm{dur}_i(q,p,\overline{d}\,) \,\Big\}$$

zusammen und können nun die Simulatorkapazitäts-Nebenbedingung formulieren:

NB $\forall\,(r,e) \in K \quad \big(\text{mit } \mathrm{fc}(r) = CP\big)$

$$\sum_{t\in e}\sum_{p\sim r}\sum_{q\in PU}\sum_{d\in D^2_{q,p,t}} \mathrm{W}(q,p,\overline{d}\,)\,\mathrm{sim}^{\mathrm{et}\to\mathrm{eh}}(p,\overline{d}\,)\cdot\Big(\mathrm{sim}^{4\mathrm{p}}(q,p,\overline{d}\,)\cdot\big(x^{(4)}_{p,d}+x^{(2)}_{p,d}+x^{(1)}_{p,d}\big)$$

$$+\,\mathrm{sim}^{2\mathrm{p}}(q,p,\overline{d}\,)\cdot\big(2x^{(4)}_{p,d}+x^{(2)}_{p,d}+x^{(1)}_{p,d}\big)\Big)\ \le\ |e|\,\mathrm{sim}(F(r),\overline{e}) \qquad (2.10)$$

Einheitenrechnung der Simulatorkapazitäts-Nebenbedingung

Betrachten wir die rechte Seite von Ungleichung (2.10) genauer. Wir berechnen hier die während des Intervalls e auf Flotte $F(r)$ zur Verfügung stehende Simulatorkapazität in [eh]. Dazu gehen wir von der Simulatorkapazität aus, die für den *letzten* Tag $\overline{e}$ des Intervalls gegeben ist (in der Einheit [eh]) und generalisieren diesen Wert auf alle Tage des Intervalls, das heißt wir nehmen an, genau jene Simulatorkapazität bestünde an *jedem* Tag des Intervalls. Dies ändert nichts am reinen Zahlenwert der Größe $\mathrm{sim}(F(r), \overline{e})$, sondern verändert implizit deren Einheit von [eh] zu [eh/d]. Eine solche implizite Einheitenänderung kennzeichnen wir, indem wir durch [1d] statt [d] teilen. Die in [eh/1d] umgewandelte Simulatorkapazität können wir nun mit der Länge $|e|$ des Intervalls (in der Einheit [d]) multipilizieren und erhalten, wie gewünscht, die im Intervall e zur Verfügung stehende Simulatorkapazität in der Einheit [eh].

Auf der linken Seite der Ungleichung geschieht das Gegenteil der Generalisierung. Mit dem äußeren Summenzeichen iterieren wir über alle Tage des Intervalls e. Der restliche Ausdruck berechnet für jeden solchen Tag t die Summe der Simulatorstunden, die bei allen Kursen auf alle PUs der Flotte $F(r)$, welche an jenem Tag im 2. Schulungsteil sind, *pro* Tag benötigt werden; jeder der Summanden hat also die Einheit [eh/d]. Da wir nun den speziellen Tag t betrachten, in dem all diese Kurse im 2. Schulungsteil sind, müssen wir den *pro* Tag berechneten Bedarfswert in einem Wert *am* Tag t umwandeln. Dies geschieht durch Multiplikation mit der impliziten Einheit [1d]; wie oben ändert sich der Zahlenwert des Ausdrucks dadurch nicht. Addieren wir die tageweisen Simulatorbedarfe nun über alle Tage $t \in e$, so erhalten wir den Simulatorbedarf auf Flotte $F(r)$ im Intervall e. Die gesamte Einheitenrechnung von Nebenbedingung (2.10) sieht also wie folgt aus:

$$\sum \Big([1\mathrm{d}] \cdot \sum \sum \sum [\,] \cdot [\mathrm{eh/et}] \cdot \big([\mathrm{et}/4\mathrm{p\,d}] \cdot [4\mathrm{p}] + [\mathrm{et}/2\mathrm{p\,d}] \cdot [2\mathrm{p}] \big) \Big) \;\leq\; [\mathrm{d}] \cdot [\mathrm{eh}/1\mathrm{d}].$$

2.4.4 Kursabstand

Anfangstage von Kursen auf dieselbe PU müssen einen vorgegebenen Mindestabstand einhalten.

D Die Funktion $\mathrm{dist} : PU \times T \to \mathbb{N}$ gebe für (p,t) den **Mindestabstand** [d] an, den ein auf PU p an Tag t beginnender Kurs zum Starttag des nächsten Kurses auf PU p haben muss.

Durch die Abstandsfunktion ist gleichzeitig für jeden Knoten (p,d) eine Menge von nachfolgenden Intervallen auf PU p gegeben, in denen keine weiteren Schulungen angelegt werden dürfen, *wenn* bei (p,d) schon eine angelegt ist. Diese Menge nennen wir den **Abstandsbereich** eines Knotens:

$$D^{\mathrm{post}}_{p,d} := \big\{\, e \in D_p \;:\; \overline{d} \leq \overline{e} < \overline{d} + \mathrm{dist}(p, \overline{d}\,) \,\big\}$$

Formulierungen wie "im Abstandsbereich eines Knotens darf höchstens eine Schulung angelegt werden" können im MIP nicht allein durch Gleichungen und Ungleichungen an die Kursvariablen modelliert werden. Erst die Einführung von Zusatzvariablen führt uns zum Ziel:

V Für jeden Knoten $(p,d) \in K$ sei die ganzzahlige Variable $x^{\mathrm{I}}_{p,d}$ ein **Indikator** [ ], der angibt, ob die Schulung $x^{\mathrm{C}}_{p,d}$ angelegt wird.

NB $\forall\,(p,d) \in K \;\; 0 \leq x^{\mathrm{I}}_{p,d} \leq 1$ (2.11)

Durch Nebenbedingung (2.11) wird der Indikator eine binäre Variable, kann also genau die beiden Werte 0 und 1 annehmen, womit die Bezeichnung *Indikator* gerechtfertigt ist. Hat er den Wert 0, soll keine Schulung angelegt werden dürfen, die entsprechende Kursvariable soll also gleich 0 sein. Hat er den Wert 1, soll eine Schulung angelegt werden dürfen. Diese beiden Aspekte modellieren wir durch folgende Ungleichung[3]:

[3] Diese Ungleichung ist stärker als unsere ursprüngliche Nebenbedingung (2.5) zur Traineebeschränkung in dem Sinne, dass jede Variablenbelegung, die die vorliegende Nebenbedingung erfüllt, auch die ursprüngliche Nebenbedingung erfüllt. Die ursprüngliche kann daher entfallen, um die Nebenbedingungsmatrix möglichst klein zu halten.

NB $\forall\,(p,d) \in K \;\; x^{\mathrm{C}}_{p,d} \leq \mathrm{cmax}(p,\overline{d}\,) \cdot x^{\mathrm{I}}_{p,d}$ (2.12)

Mit Hilfe der Indikatoren ist es nun ein Leichtes, die Kursabstands-Bedingungen zu formulieren: Im Abstandsbereich jedes Knotens darf höchstens ein Kurs angelegt werden, die Summe der Indikatoren des Abstandsbereiches darf also höchstens 1 sein. Einen Ausnahmefall gibt es zu beachten: Eingefrorene Kurse dürfen laut Problemstellung die Kursabstände untereinander verletzen. Dies modellieren wir, indem wir für jene Intervalle, in deren Abstandsbereich ein eingefrorener Kurs liegt oder die im Abstandsbereich eines eingefrorenen Kurses liegen,

$$D_p^{\text{near freez}} := \left\{ d \in D_p : \left(D_{p,d}^{\text{post}} \cap D_p^{\text{freez}} \neq \varnothing\right) \;\vee\; \left(\exists\, e \in D_p^{\text{freez}} \;\; d \in D_{p,e}^{\text{post}}\right) \right\}$$

keine Abstands-Nebenbedingungen formulieren, sondern die entsprechenden Kursvariablen gleich explizit auf 0 setzen.

NB $\forall\,(p,d) \in K$ (mit $d \notin D_p^{\text{freez}}$)

$$d \notin D_p^{\text{near freez}} \;:\; \sum_{e \in D_{p,d}^{\text{post}}} x^{\mathrm{I}}_{p,e} \leq 1 \tag{2.13}$$

$$d \in D_p^{\text{near freez}} \;:\; x^{\mathrm{C}}_{p,d} = 0 \tag{2.14}$$

2.5 Deltarechnung

In diesem Abschnitt werden wir für jeden Knoten eine Variable x^{eh} für das Einsatzstunden-Delta und eine Variable x^{et} für das Einsatztage-Delta einführen. Die Werte dieser Variablen werden wir vollständig durch Gleichungs-Nebenbedingungen bestimmen, indem wir vom eh-/et-Bestand jedes Knotens den eh-/et-Bedarf abziehen. Beim Bereitstellen der einzelnen Summanden der Deltas bezeichnen wir Bestände mit "res" (Ressource) und Bedarfe mit "dem" (Demand). Diese Ausdrücke sind selbst keine Variablen, sondern Zwischenergebnisse während der Modellierung (entsprechend den *Nichtterminalsymbolen* der theoretischen Informatik). Verwenden wir einen solchen Ausdruck dann in einer Nebenbedingung, steht er dort wieder für den Term, durch den er definiert wurde.

2.5.1 Bestand

Bestand durch aktive Piloten

Vom Pilotenbestand eines Knotens ist jeweils nur ein gewisser Anteil aktiv; die Übrigen sind in Mutterschutz, Erziehungsurlaub oder unbezahltem Sonderurlaub.

D Die Funktion $\mathrm{act}^{\mathrm{perc}} : PU \times T \to [0,1]$ gebe für (p,t) den **Anteil der aktiven Piloten** (active percentage) [ ] auf PU p an Tag t an.

Hinzu kommen die Ausgeliehenen einer PU; sie zählen nicht zum Pilotenbestand der PU, gehen aber in den Aktivenbestand mit ein. Umgekehrt müssen die Abgeordneten abgezogen werden, da sie zum Pilotenbestand der PU zählen, aber auf einer anderen PU Dienst tun. Auch teilzeitarbeitende Piloten müssen berücksichtigt werden, da sie ebenfalls den Aktivenbestand (um eine gebrochene Pilotenzahl) verringern. Alle hier genannten Einflüsse auf den Aktivenbestand sind absolut, d.h. unabhängig vom Pilotenbestand.

D Die Funktion $\mathrm{act}^{\mathrm{abs}} : PU \times T \to \mathbb{R}$ gebe für (p,t) den **Bestand an zusätzlichen aktiven Piloten** [p] auf PU p an Tag t an.

Der aktive Pilotenbestand eines Knotens $(p,d) \in K$ beträgt damit $\mathrm{act}^{\mathrm{perc}}(p,\overline{d}\,) \cdot x^{\mathrm{P}}_{p,d} + \mathrm{act}^{\mathrm{abs}}(p,\overline{d}\,)$. Da jedes Diskretisierungsintervall mehrere Tage zusammenfasst, steht uns diese Pilotenzahl an jedem Tag des Intervalls zur Verfügung, d.h. die aktiven Pilotentage innerhalb eines Knotens sind

$$|d| \, \Big(\, \mathrm{act}^{\mathrm{perc}}(p,\overline{d}\,) \cdot x^{\mathrm{P}}_{p,d} \, + \, \mathrm{act}^{\mathrm{abs}}(p,\overline{d}\,) \, \Big).$$

Lassen wir uns nun die in Abschnitt 1.3.1 beschriebene Produktivität der aktiven Piloten geben:

D Die Funktion $\mathrm{prod}^{\mathrm{eh}} : PU \times T \to [0,24]$ gebe für (p,t) die **Einsatzstunden-Produktivität** [eh/p] eines aktiven Piloten auf PU p am Tag t an.

D Die Funktion $\mathrm{prod}^{\mathrm{et}} : PU \times T \to [0,1]$ gebe für (p,t) die **Einsatztage-Produktivität** [et/p] eines aktiven Piloten auf PU p am Tag t an.

Damit errechnen sich die durch aktive Piloten verfügbaren Ressourcen (pres) wie folgt:

$$\forall\,(p,d) \in K \quad \mathrm{pres}^{\mathrm{eh}}_{p,d} \;:=\; \mathrm{prod}^{\mathrm{eh}}(p,\overline{d}\,) \;\; |d| \, \Big(\, \mathrm{act}^{\mathrm{perc}}(p,\overline{d}\,) \cdot x^{\mathrm{P}}_{p,d} \, + \, \mathrm{act}^{\mathrm{abs}}(p,\overline{d}\,) \, \Big)$$

$$\forall\,(p,d) \in K \quad \mathrm{pres}^{\mathrm{et}}_{p,d} \;:=\; \mathrm{prod}^{\mathrm{et}}(p,\overline{d}\,) \;\; |d| \, \Big(\, \mathrm{act}^{\mathrm{perc}}(p,\overline{d}\,) \cdot x^{\mathrm{P}}_{p,d} \, + \, \mathrm{act}^{\mathrm{abs}}(p,\overline{d}\,) \, \Big)$$

$$\text{Einheiten:} \quad [\mathrm{eh}] = [\mathrm{eh/p\,1d}] \cdot [\mathrm{d}] \cdot \big(\, [\,] \cdot [\mathrm{p}] + [\mathrm{p}] \, \big), \quad [\mathrm{et}] = [\mathrm{et/p\,1d}] \cdot [\mathrm{d}] \cdot \big(\, [\,] \cdot [\mathrm{p}] + [\mathrm{p}] \, \big)$$

Fast alle Bedarfs- und Bestandsrechnungen werden sowohl in Einsatzstunden als auch in Einsatztagen durchgeführt; die Formeln unterscheiden sich lediglich durch die Kürzel "eh" bzw. "et" in den betreffenden Termen. Wir werden daher im vorliegenden Abschnitt die Eingangsdaten und Berechnungen lediglich für Einsatzstunden angeben und mit einem * kennzeichnen, dass für Einsatztage alles analog verläuft. Sollte eine et-Rechnung abweichen, werden wir sie extra aufführen.

FO-Leistung

Während der Linienausbildung des 3. Schulungsteiles erbringen die Trainees bzw. die Checker zusätzliche FO-Leistung. Unabhängig davon, auf welche Cockpitfunktion der Kurs schult, wird immer der eh-/et-Bestand der entsprechenden FO-Planungseinheiten erhöht.

D Die Funktion $\mathrm{foline}^{\mathrm{eh}} : PU \times PU \times T \to [0,24]$ gebe bei (q,p,t) für eine an Tag t beginnende Schulung auf PU p den **durch FO-Leistung erbrachten Einsatzstunden-Bestand** [eh/p d]* pro Trainee und pro Tag des 3. Schulungsteils an.

Die in einem Intervall einer FO-PU erbrachte FO-Leistung ergibt sich als Summe der FO-Leistungen aller Kurse auf alle PUs der entsprechenden Flotte, die während des Intervalls im 3. Schulungsteil sind. Fassen wir wieder jene Intervalle von PU p, in denen Kurse starten, deren Trainees von PU q am Tag t im 3. Schulungsteil sind, in der Menge

$$D^3_{q,p,t} := \Big\{ \, d \in D_p \, : \, \overline{d} + \sum_{i=0}^{2} \mathrm{dur}_i(q,p,\overline{d}\,) \le t < \overline{d} + \sum_{i=0}^{3} \mathrm{dur}_i(q,p,\overline{d}\,) \, \Big\}$$

zusammen, so können wir die durch FO-Leistung erbrachten Ressourcen (fres) wie folgt bestimmen:

$$\forall\,(r,e) \in K \quad \mathrm{fres}^{\mathrm{eh}}_{r,e} \;\overset{*}{:=}\; [\,\text{falls } \mathrm{fc}(r) = FO\,] \sum_{t \in e} \sum_{p \sim r} \sum_{q \in PU} \sum_{d \in D^3_{q,p,t}} \mathrm{W}(q,p,\overline{d}\,) \; \mathrm{foline}^{\mathrm{eh}}(q,p,\overline{d}\,) \cdot x^{\mathrm{C}}_{p,d}$$

$$\text{Einheiten:} \quad [\mathrm{eh}] \;\overset{*}{=}\; \sum \Big([\mathrm{1d}] \cdot \sum \sum \sum [\,] \cdot [\mathrm{eh/p\,d}] \cdot [\mathrm{p}] \Big)$$

Mehrflugstunden

Mehrflugstunden werden dann benötigt, wenn der durch Pilotenressourcen erzeugte Bestand nicht ausreicht, um den Bedarf zu decken. Da der Pilotenbestand von den angelegten Kursen abhängt und damit variabel ist, legen wir auch für die Mehrflugstunden Variablen an:

V Für $(p,d) \in K$ gebe die reelle Variable $x^{\text{oh}}_{p,d}$ die **Anzahl der Mehrflugstunden** (overtime hours) [eh] an, die von den aktiven Piloten auf PU p im Intervall d erbracht werden.

Durch tarifliche und gesetzliche Regelungen ist die Anzahl der maximal erbringbaren Überstunden beschränkt. Diese Schranke hängt prozentual von den pro Pilot erbrachten Einsatzstunden ab.

D Die Funktion oh : $PU \times T \to [0,1]$ gebe für (p,t) den **Anteil des** auf PU p an Tag t **maximal erbringbaren Mehrflugstunden-Bestandes** (overtime hours percentage) [ ] am durch aktive Piloten verfügbaren Einsatzstunden-Bestand an.

NB $$\forall\,(p,d) \in K \;\; 0 \;\leq\; x^{\text{oh}}_{p,d} \;\leq\; \text{oh}(p,\overline{d}\,) \cdot \text{pres}^{\text{eh}}_{p,d} \tag{2.15}$$

Mehrflugstunden haben keine Entsprechung bei den Einsatztagen – Mehrflugtage gibt es also nicht.

2.5.2 Bedarf

Flugprogrammbedarf

Durch Auswertung historischer Angebots-Nachfrage-Verhältnisse und Berücksichtigung der zu erwartenden wirtschaftlichen und politischen Situation wird für den gesamten Strategiezeitraum der Flugprogrammbedarf der Planungseinheiten prognostiziert. Diese Prognose ist nicht Aufgabe der langfristigen Schulungsplanung, wir sehen den Flugprogrammbedarf daher als fest gegeben an.

D Die Funktion $\text{fdem}^{\text{eh}} : PU \times T \to \mathbb{R}_+$ gebe für (p,t) den auf PU p an Tag t bestehenden **Einsatzstunden-Flugprogrammbedarf** [eh]* an.

Sonstiger Bedarf

Neben dem absoluten Flugprogrammbedarf fällt sonstiger eh-/et-Bedarf an (siehe Abschnitt 1.3.2), der von den Piloten selbst verursacht wird, also von den Pilotenzahlen der PUs abhängig ist.

D Die Funktion $\text{add}^{\text{eh}} : PU \times T \to \mathbb{R}_+$ gebe für (p,t) den auf PU p an Tag t anfallenden **sonstigen Einsatzstunden-Bedarf** [eh/p]* pro Pilot an.

Der sonstige Bedarf adem (additional demand) eines Knotens $(p,d) \in K$ besteht pro Pilot und pro Tag des Intervalls, wird also berechnet durch

$$\text{adem}^{\text{eh}}_{p,d} \;\overset{*}{:=}\; \text{add}^{\text{eh}}(p,\overline{d}\,)\;|d| \cdot x^{\text{P}}_{p,d} \qquad\qquad \text{Einheiten}: \quad [\text{eh}] \;\overset{*}{=}\; [\text{eh/p 1d}] \cdot [\text{d}] \cdot [\text{p}]$$

Checker-Bedarf

Der bei diversen Aktivitäten anfallende Checker-Bedarf geht jeweils auf der entsprechenden CP-Planungseinheit als Bedarf ein. Obwohl nicht alle Piloten der CP-PUs auch wirklich Checker sind, werden wir davon ausgehen, dass zu jedem Zeitpunkt genügend viele Piloten der CP-PUs als Checker ausgebildet sind. Sollten die Checker bei der realen Durchführung eines Schulungsplanes de facto nicht ausreichen, werden kurzfristig weitere CPs zu Checkern ausgebildet. Checker-Bedarf fällt bei folgenden 3 Situationen an: während der Simulatorausbildung in Schulungen, bei routinemäßigen Nachschulungen der Piloten im Simulator/bei Linienflügen sowie durch anderen Bedarf der Checker wie Meetings, Seminare und Reisen zu den Ausbildungsorten. Der beim ersten Punkt entstehende Checker-Bedarf entspricht genau dem in Abschnitt 2.4.3 berechneten Simulatorbedarf, da für jede Simulatoreinheit genau ein Checker benötigt wird. Die beiden anderen Bedarfe stehen uns als Eingangsdaten Verfügung.

D Die Funktion $\text{rout}^{\text{eh}} : PU \times T \to [0, 24]$ gebe für (p, t) den auf PU p an Tag t bestehenden **Einsatzstunden-Bedarf für routinemäßige Nachschulungen** [eh/p]* pro Pilot an.

D Die Funktion $\text{other}^{\text{eh}} : PU \times T \to [0, 24]$ gebe für (p, t) den auf PU p an Tag t bestehenden **Einsatzstunden-Bedarf zur Checker-Ausbildung** [eh/p]* pro Pilot an und nehme auf Nicht-CP-PUs den Wert 0 an.

Die Funktion other hat auf Nicht-CP-PUs den Wert 0, da Checker nur auf CP-PUs ausgebildet werden. Dass der other-Bedarf *pro Pilot* gegeben ist, obwohl gar nicht alle Piloten einer CP-PU auch Checker sind, ist so zu verstehen, dass jeweils der Anteil der Piloten, die Checker sind, und *deren* Einsatzstunden-Bedarf zur Checker-Ausbildung miteinander multipliziert werden, wodurch tatsächlich ein Bedarf pro Pilot festgeschrieben werden kann.

Setzen wir nun den gesamten Checker-Einsatzstunden-Bedarf cdem^{eh} (checker demand) als Summe der drei ermittelten Bedarfe zusammen. Um den Checker-Einsatztage-Bedarf cdem^{et} zu erhalten, entferne man in der Gleichung den Faktor $\text{sim}^{\text{et}\to\text{eh}}$ und ersetze wie üblich alle eh durch et.

$$\forall\,(r,e) \in K\ \big(f := F(r)\big) \qquad \text{cdem}^{\text{eh}}_{r,e} \;:=\; [\,\text{falls fc}(r) = CP\,] \Bigg(\text{other}^{\text{eh}}(r, \overline{e}\,)\,|e| \cdot x^{\text{P}}_{r,e}$$
$$+ \sum_{t\in e} \sum_{p\in f} \text{rout}^{\text{eh}}(p,t) \cdot x^{\text{P}}_{p,D_p(t)} \;+\; \sum_{t\in e} \sum_{p\in f} \sum_{q\in PU} \sum_{d\in D^2_{q,p,t}} \text{W}(q,p,\overline{d}\,)\ \text{sim}^{\text{et}\to\text{eh}}(f,\overline{d}\,) \cdot$$
$$\cdot \Big(\text{sim}^{4\text{p}}(q,p,\overline{d}\,) \cdot \big(x^{(4)}_{p,d} + x^{(2)}_{p,d} + x^{(1)}_{p,d}\big) + \ \text{sim}^{2\text{p}}(q,p,\overline{d}\,) \cdot \big(2x^{(4)}_{p,d} + x^{(2)}_{p,d} + x^{(1)}_{p,d}\big)\ \Big) \Bigg)$$

Einheiten der ersten beiden Summanden: $\quad [\text{eh/p}\,1\text{d}] \cdot [\text{d}] \cdot [\text{p}] + \sum\sum [\text{eh/p}] \cdot [\text{p}]$

TraFO-Bedarf

D Die Funktion $\text{trafo}^{\text{eh}} : PU \times PU \times T \to \mathbb{R}_+$ gebe bei (q, p, t) für von PU q stammende Trainees einer an Tag t beginnenden Schulung auf PU p den **Bedarf an TraFO-Einsatzstunden** [eh/4p d]* pro Pilotenquadrupel und pro Tag des 1. Schulungsteils an.

Wir fassen jene Intervalle von PU p, in denen Kurse starten, deren Trainees von PU q am Tag t im 1. Schulungsteil sind, in der Menge

$$D^1_{q,p,t} := \Big\{\, d \in D_p \,:\, \overline{d} + \text{dur}_0(q,p,\overline{d}\,) \le t < \overline{d} + \sum_{i=0}^{1} \text{dur}_i(q,p,\overline{d}\,) \,\Big\}$$

zusammen und errechnen den TraFO-Bedarf tdem eines Knotens $(r, e) \in K$ wie folgt:

$$\text{tdem}^{\text{eh}}_{r,e} \;\overset{*}{:=}\; [\,\text{falls fc}(r) = FO\,] \sum_{t\in e} \sum_{p\sim r} \sum_{q\in PU} \sum_{d\in D^2_{q,p,t}} \text{W}(q,p,\overline{d}\,)\ \text{trafo}^{\text{eh}}(q,p,\overline{d}\,) \cdot \big(x^{(4)}_{p,d} + x^{(2)}_{p,d} + x^{(1)}_{p,d}\big)$$

$$\text{Einheiten:} \quad [\text{eh}] \;\overset{*}{=}\; \sum \Big([1\text{d}] \cdot \sum\sum\sum [\,] \cdot [\text{eh/4p d}] \cdot [4\text{p}] \Big)$$

2.5.3 Delta

Schließlich definieren wir für jeden Knoten die beiden eingangs besprochenen Deltavariablen und berechnen ihre Werte jeweils als Differenz aller aufgeführten Bestände und Bedarfe.

V Für jeden Knoten $(p, d) \in K$ geben die reellen Variablen $x^{\text{eh}}_{p,d}$ und $x^{\text{et}}_{p,d}$ das **Einsatzstunden-Delta** [eh] und das **Einsatztage-Delta** [et] auf PU p im Intervall d an.

NB $\forall\,(p, d) \in K$

$$x^{\text{eh}}_{p,d} \;=\; \text{pres}^{\text{eh}}_{p,d} + \text{fres}^{\text{eh}}_{p,d} + x^{\text{oh}}_{p,d} - \text{fdem}^{\text{eh}}(p,\overline{d}\,) - \text{adem}^{\text{eh}}_{p,d} - \text{cdem}^{\text{eh}}_{p,d} - \text{tdem}^{\text{eh}}_{p,d} \qquad (2.16)$$

$$x^{\text{et}}_{p,d} \;=\; \text{pres}^{\text{et}}_{p,d} + \text{fres}^{\text{et}}_{p,d} \qquad - \text{fdem}^{\text{et}}(p,\overline{d}\,) - \text{adem}^{\text{et}}_{p,d} - \text{cdem}^{\text{et}}_{p,d} - \text{tdem}^{\text{et}}_{p,d} \qquad (2.17)$$

2.6 Kosten

In den vergangenen Abschnitten haben wir die Bedingungen für die Zulässigkeit eines Schulungsplanes formuliert. Nun wollen wir uns daran machen, verschiedene zulässige Schulungspläne miteinander hinsichtlich ihrer Güte zu vergleichen. Als Gütekriterien haben wir in Abschnitt 1.3 das qualitative Ziel eines Deltas in der Nähe von 0 und das monetäre Ziel geringer Schulungskosten herausgestellt. Um beide Aspekte zu vereinen und eine eindeutige Beurteilung von Schulungsplänen zu gewährleisten, werden wir beide Ziele mittels Kosten ausdrücken. Dies geschieht im MIP durch die Zielfunktion: Wir ordnen jedem Schulungsplan einen Wert zu, der die durch ihn verursachten Kosten angibt. Unser Ziel ist es dann, einen Schulungsplan geringer Kosten zu finden.

2.6.1 Deltakosten

Betrachten wir das Einsatzstunden-Delta $x^{\text{eh}}_{p,d}$ eines Knotens (p,d). Es gibt uns die Überdeckung bzw. Unterdeckung an Einsatzstunden im Intervall d an. Mittels des Produktivitätsfaktors rechnen wir diese Einsatzstunden in eine entsprechende Zahl von Piloten um:

$$\frac{1}{\text{prod}^{\text{eh}}(p,\overline{d}\,)} \cdot x^{\text{eh}}_{p,d} \qquad \text{Einheiten}: \ \frac{1}{[\text{eh/p 1d}]} \cdot [\text{eh}].$$

Überschüssige Piloten verursachen, hauptsächlich aufgrund ihrer Gehälter, zusätzliche Kosten. Um die von einem fehlenden Piloten verursachten Kosten abzuschätzen, können wir beispielsweise die Crewstärke, die nötig ist, um ein Flugzeug einer bestimmten Flotte an einem Tag zu bereedern, und die Kosten für das Stehenlassen eines solchen Flugzeuges verwenden. Im Allgemeinen sind Unterdeckungskosten wesentlich höher als Überdeckungskosten. Die genaue Berechnungsformel beider Kosten ist für uns nicht relevant, sie seien uns einfach pro Pilot und Tag geben.

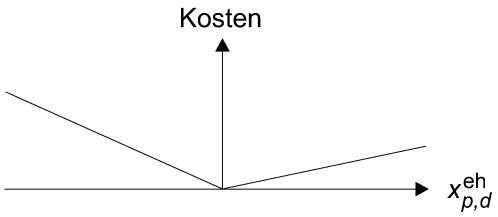
D Die Funktion high : $PU \times T \to \mathbb{R}_+$ gebe für (p,t) die an Tag t von einem überschüssigen Piloten auf PU p verursachten **Überdeckungskosten** [€/p] an.

D Die Funktion low : $PU \times T \to \mathbb{R}_+$ gebe für (p,t) die an Tag t von einem fehlenden Piloten auf PU p verursachten **Unterdeckungskosten** [€/p] an.

Die vom eh-Delta eines Knotens (p,d) verursachten Kosten berechnen sich damit wie folgt:

$$\text{Kosten} = \begin{cases} +\dfrac{\text{high}(p,\overline{d}\,)}{\text{prod}^{\text{eh}}(p,\overline{d}\,)} \cdot x^{\text{eh}}_{p,d} & \text{falls } x^{\text{eh}}_{p,d} \geq 0 \\ -\dfrac{\text{low}(p,\overline{d}\,)}{\text{prod}^{\text{eh}}(p,\overline{d}\,)} \cdot x^{\text{eh}}_{p,d} & \text{falls } x^{\text{eh}}_{p,d} < 0 \end{cases}$$

$$\text{Einheiten}: \ [€] = \pm\frac{[€/\text{p 1d}]}{[\text{eh/p 1d}]} \cdot [\text{eh}]$$



Durch die Fallunterscheidung $x^{\text{eh}}_{p,d} \geq 0$ bzw. $x^{\text{eh}}_{p,d} < 0$ wird die Kostenfunktion stückweise linear. Im MIP können wir eine solche Funktion durch Einführen einer weiteren Variablen realisieren:

V Für $(p,d) \in K$ gebe die reelle Variable $x^{€}_{p,d}$ die **Deltakosten** [€] bei Knoten (p,d) an.

Diese Variable soll nun genau den Wert der angegebenen stückweise linearen Funktion annehmen. Um dieses Ziel zu erreichen, sorgen wir erstens mit zwei Ungleichungs-Nebenbedingungen dafür, dass die Variable *nirgends unterhalb* der stückweise linearen Funktion liegt, und nehmen zweitens die Variable mit dem positiven Koeffizienten 1 in die Zielfunktion unseres Minimierungsproblems auf, wodurch sie bei jeder optimalen Lösung des MIP immer *genau* den Wert der stückweise linearen Funktion annimmt.

NB $\forall (p,d) \in K \quad x^{€}_{p,d} \geq -\frac{\text{low}(p,\overline{d})}{\text{prod}^{\text{eh}}(p,\overline{d})} \cdot x^{\text{eh}}_{p,d}$ (2.18)

NB $\forall (p,d) \in K \quad x^{€}_{p,d} \geq +\frac{\text{high}(p,\overline{d})}{\text{prod}^{\text{eh}}(p,\overline{d})} \cdot x^{\text{eh}}_{p,d}$ (2.19)

ZF $\forall (p,d) \in K$ habe die Variable $x^{€}_{p,d}$ den Zielfunktionskoeffizient 1.

Durch die stückweise lineare Kostenfunktion steigen die Kosten in gleichem Maße an wie das Delta. Sollte es stattdessen wünschenswert sein, die Abweichung vom Nulldelta immer stärker werdend zu bestrafen, kann alternativ auch eine quadratische Zielfunktion gewählt werden, wodurch sich das zugrundeliegende Modell zu einem *Mixed Integer Quadratic Program* (MIQP) [2] ändert.

Beim anderen Delta-Kriterium, dem *Einsatztage*-Delta, werden aus gleichen Gründen wie beim Einsatzstunden-Delta ebenfalls Kosten durch Unter- und durch Überdeckung erzeugt. Insgesamt entstehen also Kosten sowohl aufgrund des eh-Deltas als auch aufgrund des et-Deltas; unser Ziel ist es, beide Kosten *gleichzeitig* zu minimieren. Eine Möglichkeit dazu ist, die *Summe* beider Kostenpunkte zu minimieren, indem wir zwei verschiedene Deltakosten-Variablen benutzen und beide in die Zielfunktion aufnehmen. Da eh-Delta und et-Delta aber eng zusammenhängen und in der Realität eher der größere beider Kostenpunkte als beide gemeinsam anfällt, werden wir die Deltakosten als das *Maximum* über die beiden durch eh-Delta und et-Delta verursachten Kosten modellieren. Dafür genügt es, folgende beiden weiteren Nebenbedingungen aufzustellen:

NB $\forall (p,d) \in K \quad x^{€}_{p,d} \geq -\frac{\text{low}(p,\overline{d})}{\text{prod}^{\text{et}}(p,\overline{d})} \cdot x^{\text{et}}_{p,d}$ (2.20)

NB $\forall (p,d) \in K \quad x^{€}_{p,d} \geq +\frac{\text{high}(p,\overline{d})}{\text{prod}^{\text{et}}(p,\overline{d})} \cdot x^{\text{et}}_{p,d}$ (2.21)

In dem Fall, dass eine der beiden Produktivitätszahlen und damit die Nenner der entsprechenden beiden Nebenbedingungen den Wert 0 annehmen, lassen wir besagte Nebenbedingungen entfallen und berechnen die Deltakosten lediglich über die beiden anderen Nebenbedingungen. Sind sogar beide Produktivitäten gleich 0, interpretieren wir dies so, dass auf der PU am entsprechenden Tag keine Deltakosten berechnet werden sollen und setzen die Kostenvariable $x^{€}_{p,d}$ explizit auf 0. Dieser Ausnahmefall wird bei realen PUs natürlich nicht vorkommen, macht aber Sinn auf der NFF-PU, da hier tatsächlich keine fliegerische Produktivität herrscht. Trotzdem legen wir auch für diese PU Deltavariablen an, um die Möglichkeit zu schaffen, durch Einführung eines künstlichen Bedarfs die Verwendung von NFFs zu steuern (zu begünstigen bei NFF-Überfluss/zu bestrafen bei Mangel).

2.6.2 Schulungskosten

Weitere Kosten entstehen pro Trainee eines Kurses. Ist die Teilnehmerzahl des Kurses außerdem ungerade, entstehen durch den *ungeraden Trainee* zusätzliche Kosten, da viele Schulungsressourcen nur paarweise genutzt werden können.

D Die Funktion tcost : $PU \times PU \times T \to \mathbb{R}_+$ gebe bei (q,p,t) für einen an Tag t beginnenden Kurs auf PU p die **Schulungskosten** (type-rating costs) [€/p] pro Trainee von PU q an.

D Die Funktion uperc : $PU \times PU \times T \to [0,1]$ gebe bei (q,p,t) für einen an Tag t beginnenden Kurs auf PU p den **Anteil der zusätzlichen Schulungskosten** (uneven type-rating costs percentage) [ ] an, die ein ungerader Trainee von PU q verursacht.

Die bei einem Kurs zu erwartenden Stamm-PUs sind uns durch die Stammverteilung gegeben, wir berechnen also die pro Trainee zu erwartenden Schulungskosten (expected type-rating costs) und die Kosten bei Teilnahme eines ungeraden Trainees (expected uneven type-rating costs) über:

$$\text{etcost}(p,t) := \sum_{q \in PU} \text{W}(q,p,t) \cdot \text{tcost}(q,p,t)$$

$$\text{eutcost}(p,t) := \sum_{q \in PU} \text{W}(q,p,t) \cdot \text{tcost}(q,p,t) \cdot \big(1 + \text{uperc}(q,p,t)\big)$$

Übernehmen wir die berechneten Kosten in die Zielfunktion. Durch jedes Traineequadrupel $x^{(4)}$ eines Kurses entstehen vierfache Type-Rating-Kosten etcost, durch das Traineepaar $x^{(2)}$ zweifache. Ein weiterer einzelner Trainee $x^{(1)}$ macht die Teilnehmerzahl ungerade und verursacht damit Kosten entsprechend eutcost.

ZF $\forall\,(p,d) \in K$ habe die Variable $x^{(4)}_{p,d}$ den Zielfunktionskoeffizient $4 \cdot \text{etcost}(p, \overline{d}\,)$.

ZF $\forall\,(p,d) \in K$ habe die Variable $x^{(2)}_{p,d}$ den Zielfunktionskoeffizient $2 \cdot \text{etcost}(p, \overline{d}\,)$.

ZF $\forall\,(p,d) \in K$ habe die Variable $x^{(1)}_{p,d}$ den Zielfunktionskoeffizient $\text{eutcost}(p, \overline{d}\,)$.

2.6.3 Überstundenkosten

Wir haben mit den Mehrflugstunden-Variablen x^{oh} die Möglichkeit geschaffen, in der Optimierung bewusst Überstunden einzuplanen, um den Einsatzstunden-Bestand zu erhöhen und das Delta zu verbessern. Dass Überstunden aber auch Kosten in Form von zusätzlichen Pilotengehältern verursachen, müssen wir in der Zielfunktion berücksichtigen.

D Die Funktion $\text{ocost} : PU \times T \to [0,1]$ gebe für (p,t) die **Kosten einer Mehrflugstunde** (overtime hours costs) [€/eh] auf PU p am Tag t an.

ZF $\forall\,(p,d) \in K$ habe die Variable $x^{\text{oh}}_{p,d}$ den Zielfunktionskoeffizient $\text{ocost}(p, \overline{d}\,)$.

2.7 Zusammenfassung

In diesem Abschnitt wollen wir alle definierten Variablen, Nebenbedingungen und Zielfunktionskoeffizienten von SPTS noch einmal in einer kompakten Form zusammenfassen, in der das auf Seite 8 definierte allgemeine MIP deutlich wiederzuerkennen ist. Die Terme, für die während der Deltarechnung Nichtterminalsymbole eingeführt wurden, werden in den entsprechenden Formeln wieder explizit eingesetzt. Gerade die Gleichungs-Nebenbedingungen zur Berechnung der Deltas (Abschnitt 2.5.3) sehen dadurch deutlich umfangreicher aus; trotzdem sind alle Terme natürlich weiterhin Linearkombinationen der angelegten Variablen.

Klären wir vor der zusammenfassenden Formulierung von SPTS noch kurz folgende Frage: Warum stellen wir die Gleichungs-Nebenbedingungen (GNB) nicht jeweils nach einer der darin vorkommenden Variablen um und setzen diesen Term anstelle der Variablen in alle Nebenbedingungen ein, in denen die Variable vorkommt? Könnte man mit diesem Verfahren, sukzessive auf alle GNB angewendet, nicht sowohl eine große Zahl an Variablen als auch alle GNB einsparen? Das könnte man in der Tat, allerdings zu dem Preis, dass die Nebenbedingungs-Matrizen des MIP dadurch dichter besetzt wären (weniger 0-Einträge hätten), was im Allgemeinen zu wesentlichen Laufzeiteinbußen bei der Lösung des Problems führen würde. Verdeutlichen wir uns dies anhand der Pilotenvariablen $x^{\text{P}}_{p,d}$. Der Wert jeder Pilotenvariable wird durch ihre entsprechende Flussgleichung (Seite 15) eindeutig bestimmt. Verzichteten wir auf diese Nebenbedingung und ersetzten stattdessen jede Pilotenvariable durch die rechte Seite ihrer Flussgleichung (nennen wir sie das *Substitut*), so enthielte das Substitut der zweiten Pilotenvariable einer PU zusätzlich das gesamte Substitut der ersten Pilotenvariable, da die erste Variable in die rechte Seite der zweiten Flussgleichung eingeht. Entsprechend enthielte das Substitut der dritten Pilotenvariable die Substitute der beiden vorherigen Pilotenvariablen, das der letzten Pilotenvariable sogar die Substitute aller anderen Pilotenvariablen der PU. In jeder Nebenbedingung, in der eine Pilotenvariable verwendet wird (was jeweils bloß einen Matrixeintrag verursacht), müssten wir nun das gesamte Substitut dieser Variable einsetzen. Der dadurch verursachte Anstieg der Matrixdichte ließe sich durch das Einsparen der Flussgleichungen in keinem Maße ausgleichen. Dieser Argumentation folgend werden wir keine der Gleichungs-Nebenbedingungen eliminieren, da alle vorkommenden Variablen noch mehr als einmal in anderen Nebenbedingungen verwendet werden.

SPTS

$$\min \Bigg\{ \sum_{(p,d)\in K} \Big(x^{€}_{p,d} + 4 \cdot \Big(\sum_{q\in PU} \mathrm{W}(q,p,t) \cdot \mathrm{tcost}(q,p,t) \Big) \cdot x^{(4)}_{p,d} + 2 \cdot \Big(\sum_{q\in PU} \mathrm{W}(q,p,t) \cdot \mathrm{tcost}(q,p,t) \Big) \cdot x^{(2)}_{p,d} + \Big(\sum_{q\in PU} \mathrm{W}(q,p,t) \cdot \mathrm{tcost}(q,p,t) \cdot \big(1 + \mathrm{uperc}(q,p,t) \big) \Big) \cdot x^{(1)}_{p,d} + \mathrm{ocost}(p,\overline{d}\,) \cdot x^{\mathrm{oh}}_{p,d} \Big) : \Bigg.$$

$\forall\, (p,d) \in K$

$$x^{\mathrm{P}}_{p,d},\ x^{\mathrm{oh}}_{p,d},\ x^{\mathrm{eh}}_{p,d},\ x^{\mathrm{et}}_{p,d},\ x^{€}_{p,d} \in \mathbb{R}$$
$$x^{\mathrm{I}}_{p,d},\ x^{\mathrm{C}}_{p,d},\ x^{(4)}_{p,d},\ x^{(2)}_{p,d},\ x^{(1)}_{p,d} \in \mathbb{Z}$$

$$0 \le x^{\mathrm{I}}_{p,d} \tag{2.11}$$
$$x^{\mathrm{I}}_{p,d} \le 1 \tag{2.11}$$
$$0 \le x^{\mathrm{C}}_{p,d} \tag{2.4}$$
$$x^{\mathrm{C}}_{p,d} \le \mathrm{cmax}(p,\overline{d}\,) \cdot x^{\mathrm{I}}_{p,d} \tag{2.12}$$
$$0 \le x^{(2)}_{p,d} \tag{2.8}$$
$$x^{(2)}_{p,d} \le 1 \tag{2.8}$$
$$0 \le x^{(1)}_{p,d} \tag{2.8}$$
$$x^{(1)}_{p,d} \le 1 \tag{2.8}$$
$$x^{\mathrm{C}}_{p,d} = 4x^{(4)}_{p,d} + 2x^{(2)}_{p,d} + x^{(1)}_{p,d} \tag{2.9}$$
$$0 \le x^{\mathrm{P}}_{p,d} \tag{2.2}$$
$$0 \le x^{\mathrm{oh}}_{p,d} \tag{2.15}$$
$$x^{\mathrm{oh}}_{p,d} \le \mathrm{oh}(p,\overline{d}\,) \cdot \mathrm{prod}^{\mathrm{eh}}(p,\overline{d}\,)\ \ |d|\ \Big(\mathrm{act}^{\mathrm{perc}}(p,\overline{d}\,) \cdot x^{\mathrm{P}}_{p,d}\ +\ \mathrm{act}^{\mathrm{abs}}(p,\overline{d}\,) \Big) \tag{2.15}$$

$$x^{\mathrm{P}}_{p,d}\ =\ [\,\text{falls } \exists\, \mathrm{pre}_p d\,]\ \mathrm{stay}(p,\overline{d}\,)^{\,|d|} \cdot x^{\mathrm{P}}_{p,\,\mathrm{pre}_p d}\ +\ \sum_{t\in d} \mathrm{inc}(p,t)\ -\ \sum_{q\in PU} \sum_{e\in D^{\mathrm{out}}_{q,p,d}} \mathrm{W}(p,q,\overline{e}\,) \cdot x^{\mathrm{C}}_{q,e}$$
$$+\ \sum_{q\in PU} \sum_{e\in D^{\mathrm{in}}_{p,d}} \big(1-\mathrm{fail}(q,p,\overline{e})\big) \cdot \mathrm{W}(q,p,\overline{e}\,) \cdot x^{\mathrm{C}}_{p,e} + \sum_{q\in PU} \sum_{e\in D^{\mathrm{in}}_{q,d}} \mathrm{fail}(p,q,\overline{e}) \cdot \mathrm{W}(p,q,\overline{e}\,) \cdot x^{\mathrm{C}}_{q,e} \tag{2.6}$$

$$(d \notin D^{\mathrm{freez}}_p) \implies \begin{cases} (d \notin D^{\text{near freez}}_p) \implies \displaystyle\sum_{e\in D^{\mathrm{post}}_{p,d}} x^{\mathrm{I}}_{p,e} \le 1 & (2.13) \\ (d \in D^{\text{near freez}}_p) \implies x^{\mathrm{C}}_{p,d} = 0 & (2.14) \end{cases}$$

$$\big(\exists\, q \in PU\ \ \mathrm{freez}(q,p,t) > 0\big) \implies x^{\mathrm{C}}_{p,\{t\}} = \sum_{q\in PU} \mathrm{freez}(q,p,t) \tag{2.7}$$

$$\big(\mathrm{fc}(p) = CP\big) \implies \sum_{t\in d}\sum_{r\sim p}\sum_{q\in PU}\sum_{e\in D^2_{q,r,t}} \mathrm{W}(q,r,\overline{e})\ \mathrm{sim}^{\mathrm{et}\to\mathrm{eh}}(r,\overline{e})\ \cdot \tag{2.10}$$

$$\Big(\mathrm{sim}^{4\mathrm{p}}(q,r,\overline{e})\cdot\big(x^{(4)}_{r,e}+x^{(2)}_{r,e}+x^{(1)}_{r,e}\big) + \mathrm{sim}^{2\mathrm{p}}(q,r,\overline{e})\cdot\big(2x^{(4)}_{r,e}+x^{(2)}_{r,e}+x^{(1)}_{r,e}\big)\Big) \ \le\ |d|\ \mathrm{sim}(F(p),\overline{d}\,)$$

$$\begin{aligned} x^{\mathrm{eh}}_{p,d} \ =\ & \mathrm{prod}^{\mathrm{eh}}(p,\overline{d}\,)\ |d|\ \Big(\mathrm{act}^{\mathrm{perc}}(p,\overline{d}\,)\cdot x^{\mathrm{P}}_{p,d} + \mathrm{act}^{\mathrm{abs}}(p,\overline{d}\,)\Big) + x^{\mathrm{oh}}_{p,d} \\ & + [\,\text{falls } \mathrm{fc}(p) = FO\,] \sum_{t\in d}\sum_{r\sim p}\sum_{q\in PU}\sum_{e\in D^3_{q,r,t}} \mathrm{W}(q,r,\overline{e})\ \mathrm{foline}^{\mathrm{eh}}(q,r,\overline{e})\cdot x^{\mathrm{C}}_{r,e} \\ & - \mathrm{fdem}^{\mathrm{eh}}(p,\overline{d}\,) - \mathrm{add}^{\mathrm{eh}}(p,\overline{d}\,)\ |d|\cdot x^{\mathrm{P}}_{p,d} - [\,\text{falls } \mathrm{fc}(p) = CP\,]\ \Big(\mathrm{other}^{\mathrm{eh}}(p,\overline{d}\,)\,|d|\cdot x^{\mathrm{P}}_{p,d} \\ & \quad + \sum_{t\in d}\sum_{r\sim p}\mathrm{rout}^{\mathrm{eh}}(r,t)\cdot x^{\mathrm{P}}_{r,D_r(t)} + \sum_{t\in d}\sum_{r\sim p}\sum_{q\in PU}\sum_{e\in D^2_{q,r,t}} \mathrm{W}(q,r,\overline{e})\ \mathrm{sim}^{\mathrm{et}\to\mathrm{eh}}(r,\overline{e})\cdot \\ & \qquad \Big(\mathrm{sim}^{4\mathrm{p}}(q,r,\overline{e})\cdot\big(x^{(4)}_{r,e}+x^{(2)}_{r,e}+x^{(1)}_{r,e}\big) + \mathrm{sim}^{2\mathrm{p}}(q,r,\overline{e})\cdot\big(2x^{(4)}_{r,e}+x^{(2)}_{r,e}+x^{(1)}_{r,e}\big)\Big)\Big) \\ & - [\,\text{falls } \mathrm{fc}(p) = FO\,] \sum_{t\in d}\sum_{r\sim p}\sum_{q\in PU}\sum_{e\in D^2_{q,r,t}} \mathrm{W}(q,r,\overline{e})\ \mathrm{trafo}^{\mathrm{eh}}(q,r,\overline{e})\cdot\big(x^{(4)}_{r,e}+x^{(2)}_{r,e}+x^{(1)}_{r,e}\big) \end{aligned} \tag{2.16}$$

$$\begin{aligned} x^{\mathrm{et}}_{p,d} \ =\ & \mathrm{prod}^{\mathrm{et}}(p,\overline{d}\,)\ |d|\ \Big(\mathrm{act}^{\mathrm{perc}}(p,\overline{d}\,)\cdot x^{\mathrm{P}}_{p,d} + \mathrm{act}^{\mathrm{abs}}(p,\overline{d}\,)\Big) \\ & + [\,\text{falls } \mathrm{fc}(p) = FO\,] \sum_{t\in d}\sum_{r\sim p}\sum_{q\in PU}\sum_{e\in D^3_{q,r,t}} \mathrm{W}(q,r,\overline{e})\ \mathrm{foline}^{\mathrm{et}}(q,r,\overline{e})\cdot x^{\mathrm{C}}_{r,e} \\ & - \mathrm{fdem}^{\mathrm{eh}}(p,\overline{d}\,) - \mathrm{add}^{\mathrm{et}}(p,\overline{d}\,)\ |d|\cdot x^{\mathrm{P}}_{p,d} - [\,\text{falls } \mathrm{fc}(p) = CP\,]\ \Big(\mathrm{other}^{\mathrm{et}}(p,\overline{d}\,)\,|d|\cdot x^{\mathrm{P}}_{p,d} \\ & \quad + \sum_{t\in d}\sum_{r\sim p}\mathrm{rout}^{\mathrm{et}}(r,t)\cdot x^{\mathrm{P}}_{r,D_r(t)} + \sum_{t\in d}\sum_{r\sim p}\sum_{q\in PU}\sum_{e\in D^2_{q,r,t}} \mathrm{W}(q,r,\overline{e})\cdot \\ & \qquad \Big(\mathrm{sim}^{4\mathrm{p}}(q,r,\overline{e})\cdot\big(x^{(4)}_{r,e}+x^{(2)}_{r,e}+x^{(1)}_{r,e}\big) + \mathrm{sim}^{2\mathrm{p}}(q,r,\overline{e})\cdot\big(2x^{(4)}_{r,e}+x^{(2)}_{r,e}+x^{(1)}_{r,e}\big)\Big)\Big) \\ & - [\,\text{falls } \mathrm{fc}(p) = FO\,] \sum_{t\in d}\sum_{r\sim p}\sum_{q\in PU}\sum_{e\in D^2_{q,r,t}} \mathrm{W}(q,r,\overline{e})\ \mathrm{trafo}^{\mathrm{et}}(q,r,\overline{e})\cdot\big(x^{(4)}_{r,e}+x^{(2)}_{r,e}+x^{(1)}_{r,e}\big) \end{aligned} \tag{2.17}$$

$$\big(\mathrm{prod}^{\mathrm{eh}}(p,\overline{d}\,) > 0\big) \implies x^{€}_{p,d} \ge -\frac{\mathrm{low}(p,\overline{d}\,)}{\mathrm{prod}^{\mathrm{eh}}(p,\overline{d}\,)}\cdot x^{\mathrm{eh}}_{p,d} \tag{2.18}$$

$$x^{€}_{p,d} \ge +\frac{\mathrm{high}(p,\overline{d}\,)}{\mathrm{prod}^{\mathrm{eh}}(p,\overline{d}\,)}\cdot x^{\mathrm{eh}}_{p,d} \tag{2.19}$$

$$\big(\mathrm{prod}^{\mathrm{et}}(p,\overline{d}\,) > 0\big) \implies x^{€}_{p,d} \ge -\frac{\mathrm{low}(p,\overline{d}\,)}{\mathrm{prod}^{\mathrm{et}}(p,\overline{d}\,)}\cdot x^{\mathrm{et}}_{p,d} \tag{2.20}$$

$$x^{€}_{p,d} \ge +\frac{\mathrm{high}(p,\overline{d}\,)}{\mathrm{prod}^{\mathrm{et}}(p,\overline{d}\,)}\cdot x^{\mathrm{et}}_{p,d} \tag{2.21}$$

$$\big(\mathrm{prod}^{\mathrm{eh}}(p,\overline{d}\,) = 0\ \wedge\ \mathrm{prod}^{\mathrm{et}}(p,\overline{d}\,) = 0\big) \implies x^{€}_{p,d} = 0 \Big\}$$

Kapitel 3

Komplexität

In diesem Kapitel werden wir nachweisen, dass das zu SPTS gehörige Entscheidungsproblem in der Komplexitätsklasse der NP-vollständigen Probleme liegt. Für eine exakte Einführung der NP-Theorie wird der Begriff der nichtdeterministischen Turingmaschine benötigt; da dies den Rahmen dieser Arbeit sprengen würde, sei auf die gute Darstellung der NP-Vollständigkeitstheorie in [7] verwiesen. Wir wollen jedoch die wichtigen Begriffe zumindest intuitiv klar machen.

Entscheidungsprobleme sind Probleme, die entweder mit *ja* oder mit *nein* beantwortet werden können. Ein Beispiel für ein Entscheidungsproblem ist PARTITION, bei dem n Zahlen $a_1, \ldots, a_n$ gegeben sind und wir uns dafür interessieren, ob die Zahlen so in zwei Teile zerlegt werden können, dass die Summe beider Teile gleich groß ist.

PARTITION

gegeben: $n \in \mathbb{N}, a \in \mathbb{N}^n$

gefragt: $\exists\, J \subseteq N := \{1, \ldots, n\} \quad \sum_{i \in J} a_i = \sum_{i \in N \setminus J} a_i \quad ?$

Betrachten wir ein bestimmtes Problem für konkrete Eingangswerte, dann sprechen wir von einer **Instanz** des Problems. Das folgende Beispiel zeigt eine Instanz des Problems PARTITION:

■ $n = 9, a = (9, 12, 11, 1, 20, 18, 9, 14, 8) \implies$ Antwort *ja*, denn 9+12+1+20+9 = 11+18+14+8

Die Komplexitätstheorie teilt Entscheidungsprobleme hinsichtlich der **Rechenzeit** ein, in der die Antwort *ja* bzw. *nein* gefunden werden kann. Für viele Probleme kennt man *effiziente* Algorithmen, d.h. Algorithmen, deren Rechenzeit durch ein Polynom, abhängig von der Codierungslänge der Instanz, beschränkt ist. Die Klasse dieser Probleme bezeichnet man mit **P**. Für viele Probleme sind jedoch (noch?) keine polynomiellen Algorithmen bekannt, aber wir können, wenn uns eine Lösung gegeben ist, in polynomieller Zeit verifizieren, dass diese tatsächlich eine Lösung ist. Jene Klasse von Problemen bezeichnet man mit **NP** (nichtdeterministisch polynomiell). PARTITION beispielsweise liegt in NP, denn ist uns eine Aufteilung der Zahlen in zwei Partitionen gegeben, müssen wir lediglich die Zahlen jeder Partition addieren und die beiden Ergebnisse vergleichen. Dieser Vorgang benötigt $n-2$ Additionen und 1 Vergleich und ist daher polynomiell durchführbar.

Innerhalb der Klasse NP gibt es besonders schwere, sogenannte **NP-vollständige Probleme**: Findet man für nur ein einziges NP-vollständiges Problem einen polynomiellen Algorithmus, dann existiert für *jedes* beliebige Problem aus NP ein polynomieller Algorithmus. Man glaubt allgemein nicht daran, dass es gelingen wird, einen polynomiellen Algorithmus für ein NP-vollständiges Problem zu finden. Der Nachweis der NP-Vollständigkeit eines gewissen Problems bedeutet also, dass die Existenz von effizienten Algorithmen für das Problem wahrscheinlich ausgeschlossen ist. Um zu zeigen, dass auch SPTS NP-vollständig ist, müssen wir glücklicherweise nicht zeigen, dass aus seiner polynomiellen Lösbarkeit die polynomielle Lösbarkeit aller NP-Probleme folgt, sondern können die Tatsache benutzen, dass bereits Probleme als NP-vollständig nachgewiesen wurden.

Die Idee ist die folgende: Können wir (a) zeigen, dass SPTS in NP liegt und finden wir (b) eine polynomielle Transformation ϕ, die jede Instanz eines nachgewiesen NP-vollständigen Problems auf eine SPTS-Instanz abbildet, so dass beide Instanzen lösungsäquivalent sind (d.h. entweder beide die Antwort *ja* oder beide die Antwort *nein* haben), so ist SPTS auch NP-vollständig. Denn hätten wir einen polynomiellen Algorithmus für SPTS, so hätten wir durch die polynomielle Transformation ϕ auch einen polynomiellen Algorithmus für das ursprüngliche NP-vollständige Problem und damit polynomielle Algorithmen für alle Probleme aus NP.

Da die NP-Vollständigkeitstheorie Aussagen über Entscheidungsprobleme trifft, wandeln wir das Optimierungsproblem SPTS in eine Klasse von Entscheidungsproblemen um. Wir fragen damit nicht mehr nach dem optimalen Zielfunktionswert z einer SPTS-Instanz, sondern danach, ob es in der Instanz einen Schulungsplan $x \in X$ gibt, der einen nicht schlechteren Zielfunktionswert cx als eine vorgegebene Zahl u hat.

$\mathbf{SPTS}_u$ $\quad (u \in \mathbb{Q}_+)$

gegeben: SPTS-Instanz

gefragt: $\exists\, x \in X \quad cx \leq u \quad ?$

Wir werden im Folgenden die NP-Vollständigkeit des speziellen Problems SPTS_0 zeigen. Man überzeugt sich leicht davon, dass auch die anderen Entscheidungsprobleme SPTS_u mit $u > 0$ NP-vollständig sind. Ist jedes dieser Entscheidungsprobleme allein schon NP-vollständig, so ist das Optimierungsproblem SPTS selbst natürlich noch schwerer lösbar.

Um für SPTS_0 Bedingung (a) einzusehen, überlegen wir uns, dass für jede Variablenbelegung x polynomiell überprüfbar ist, ob die Nebenbedingungen von SPTS erfüllt sind (also ob x ein Schulungsplan ist) und ob die Ungleichung $cx \leq 0$ erfüllt ist (also ob x Lösung von SPTS_0 ist). Für einen Schulungsplan $x \in X$ mit $cx \leq 0$ können wir demnach polynomiell verifizieren, dass er das Entscheidungsproblem SPTS_0 tatsächlich löst. Damit ist SPTS_0 in der Klasse NP enthalten.

Um (b) zu zeigen, benötigen wir eine polynomielle lösungsäquivalente Transformation von einem NP-vollständigen Problem auf SPTS_0. Als NP-vollständiges Problem werden wir PARTITION benutzen. Die folgende polynomielle Transformation ϕ wandelt jede PARTITION-Instanz (n, a) in eine SPTS_0-Instanz $\phi(n, a)$ um:

$$\phi:\ (n,a)\ \mapsto\ \begin{cases} PU = \{J_0, J_1, P_1, \ldots, P_n\},\ D = \big\{\{-1\},\{0\},\{1\},\{2\}\big\} \\ \text{stay} = \text{dur} = \text{act}^{\text{perc}} = \text{low} = \text{high} = 1, \text{dist} = 2 \\ \text{inc}(J_0, -1) = \text{inc}(J_1, -1) = \frac{1}{2}\sum a_i \\ \forall\, i \in N\ \ \text{prod}^{\text{et}}(P_i, 2) = 1,\ \text{fdem}^{\text{et}}(P_i, 2) = a_i \\ \forall\, i \in N\ \ \forall t \in \{0,1\}\ \ \text{cmax}(P_i, t) = a_i,\ \text{W}(J_t, P_i, t) = 1 \end{cases}$$

Die nicht angegebenen Werte der Stammverteilung seien konsistent zu (W_1) und (W_2) gewählt, alle anderen nicht angegebenen Eingangsdaten seien 0. Was genau tut nun diese Transformation? Betrachten wir dazu Abbildung 3.1 auf Seite 31: Für jedes a_i wird eine Planungseinheit P_i erstellt, außerdem führen wir zwei zusätzliche PUs J_0 und J_1 ein. Die Diskretisierungen aller PUs bestehen aus den 4 eintägigen Intervallen $\{-1\}, \{0\}, \{1\}$ und $\{2\}$. Die zu partitionierende Summe $\sum a_i$ wird halbiert und als Anfangspilotenbestand auf die PUs J_0 und J_1 aufgeteilt. Auf jeder PU P_i bieten wir zwei Schulungen für je maximal a_i Trainees an: eine bei $\{0\}$, deren sämtliche Trainees von PU J_0 stammen (falls sie angelegt wird), und eine bei $\{1\}$, deren sämtliche Trainees von PU J_1 stammen. Aufgrund des Mindestabstandes 2 kann auf jeder PU höchstens eine dieser beiden Schulungen angelegt werden. Am letzten Tag besteht auf den PUs P_i jeweils ein Flugprogrammbedarf von a_i. Wir zeigen nun, dass die beschriebene SPTS_0-Instanz $\phi(n, a)$ genau dann mit *ja* beantwortet wird (d.h. es einen Schulungsplan $x \in X$ mit Zielfunktionswert $cx \leq 0$ gibt), wenn die ursprüngliche PARTITION-Instanz (n, a) mit *ja* beantwortet wird (d.h. es eine Partition der Zahlen $a_1, \ldots, a_n$ in zwei Teile gibt, so dass beide Teile dieselbe Summe haben).

Satz. $\forall n \in \mathbb{N}, a \in \mathbb{N}^n$

$$\textit{PARTITION liefert für die Instanz } (n,a) \textit{ die Antwort ja} \iff$$
$$\textit{SPTS}_0 \textit{ liefert für die Instanz } \phi(n,a) \textit{ die Antwort ja}$$

Beweis $\Longrightarrow$: Liefert PARTITION für die Instanz (n,a) die Antwort *ja*, so gibt es eine Partition $J \subseteq N$ mit $\sum_{i\in J} a_i = \sum_{i\in N\setminus J} a_i$. In der SPTS$_0$-Instanz $\phi(n,a)$ erstellen wir nun einen Schulungsplan x, bei dem auf jeder PU P_i genau eine Schulung angelegt wird, und zwar bei $\{1\}$, wenn $i \in J$, und bei $\{0\}$, wenn $i \in N \setminus J$:

$$\forall\, i \in N \quad x^{\mathrm{C}}_{P_i,\{\chi_J(i)\}} := a_i$$

Der hierdurch erzeugte Schulungsplan x ist zulässig, da die angelegten Kurse die vorgegebenen Traineebeschränkungen einhalten und da die Pilotenbestände der PUs J_0 und J_1 an den Tagen, an denen Piloten von dort durch Schulungen abgezogen werden, nichtnegativ bleiben:

$$\forall\, t \in \{0,1\} \quad x^{\mathrm{P}}_{J_t,\{t\}} = x^{\mathrm{P}}_{J_t,\{-1\}} - \sum_{i\in N} x^{\mathrm{C}}_{P_i,\{t\}} = \frac{1}{2}\sum_{i\in N} a_i - \sum_{i\in J} a_i = \frac{1}{2}\sum_{i\in N} a_i - \frac{1}{2}\sum_{i\in N} a_i = 0 \geq 0$$

Ferner ist der Zielfunktionswert des Schulungsplanes x gleich 0, also liefert SPTS$_0$ die Antwort *ja*:

$$\forall\, i \in N \quad x^{\mathrm{P}}_{P_i,\{2\}} = x^{\mathrm{P}}_{P_i,\{-1\}} + x^{\mathrm{C}}_{P_i,\{0\}} + x^{\mathrm{C}}_{P_i,\{1\}} = 0 + \begin{cases} 0 + a_i : & i \in J \\ a_i + 0 : & i \in N \setminus J \end{cases} = a_i \implies$$

$$\forall\, i \in N \quad x^{\mathrm{et}}_{P_i,\{2\}} = 1 \cdot |\{2\}| \cdot (1 \cdot a_i) - a_i = 0 \implies \forall\, i \in N \quad x^{\text{€}}_{P_i,\{2\}} = 0 \implies cx = 0.$$

$\Longleftarrow$: Liefert umgekehrt SPTS$_0$ für die Instanz $\phi(n,a)$ die Antwort *ja*, so gibt es einen Schulungsplan x mit Zielfunktionswert $cx \leq 0$. Aufgrund der Nichtnegativität der Zielfunktion gilt sogar $cx = 0$. Im Umkehrung obiger Aussagenkette folgt daraus für alle $i \in N$:

$$x^{\text{€}}_{P_i,\{2\}} = 0 \implies x^{\mathrm{et}}_{P_i,\{2\}} = 0 \implies x^{\mathrm{P}}_{P_i,\{2\}} = a_i$$

Da der Anfangspilotenbestand jeder PU P_i gleich 0 ist, muss der erhöhte Pilotenbestand a_i des Intervalls $\{2\}$ von Schulungen herrühren. Wegen des Mindestabstandes 2 und der Traineebeschränkung a_i ist auf PU P_i genau eine der beiden Schulungen $x^{\mathrm{C}}_{P_i,\{0\}}$ und $x^{\mathrm{C}}_{P_i,\{1\}}$ angelegt:

$$\forall\, i \in N \quad \left(x^{\mathrm{C}}_{P_i,\{0\}} = a_i \ \wedge\ x^{\mathrm{C}}_{P_i,\{1\}} = 0\right) \ \dot\vee\ \left(x^{\mathrm{C}}_{P_i,\{0\}} = 0 \ \wedge\ x^{\mathrm{C}}_{P_i,\{1\}} = a_i\right)$$

Für die PARTITION-Instanz (n,a) konstruieren wir nun die folgende Partition J:

$$J := \{\, i \in N : \ x^{\mathrm{C}}_{P_i,\{1\}} = a_i \,\}$$

Summieren wir die a_i für $i \in J$ und für $i \in N \setminus J$ auf, so erhalten wir die beiden Ungleichungen

$$\sum_{i\in J} a_i = \sum_{i\in J} x^{\mathrm{C}}_{P_i,\{1\}} = \sum_{i\in N} x^{\mathrm{C}}_{P_i,\{1\}} \leq x^{\mathrm{P}}_{J_1,\{-1\}} = \frac{1}{2}\sum_{i\in N} a_i$$

$$\sum_{i\in N\setminus J} a_i = \sum_{i\in N\setminus J} x^{\mathrm{C}}_{P_i,\{0\}} = \sum_{i\in N} x^{\mathrm{C}}_{P_i,\{0\}} \leq x^{\mathrm{P}}_{J_0,\{-1\}} = \frac{1}{2}\sum_{i\in N} a_i$$

In keiner der beiden Ungleichungen kann ein echtes $<$ gelten, da Summation beider Seiten sonst auf einen Widerspruch führen würde. Es handelt sich also in Wirklichkeit um zwei Gleichungen und es gilt $\sum_{i\in J} a_i = \sum_{i\in N\setminus J} a_i$. Damit teilt Partition J die Zahlen $a_1, \ldots, a_n$ in zwei Teile mit jeweils gleicher Summe auf und PARTITION liefert die Antwort *ja*. □

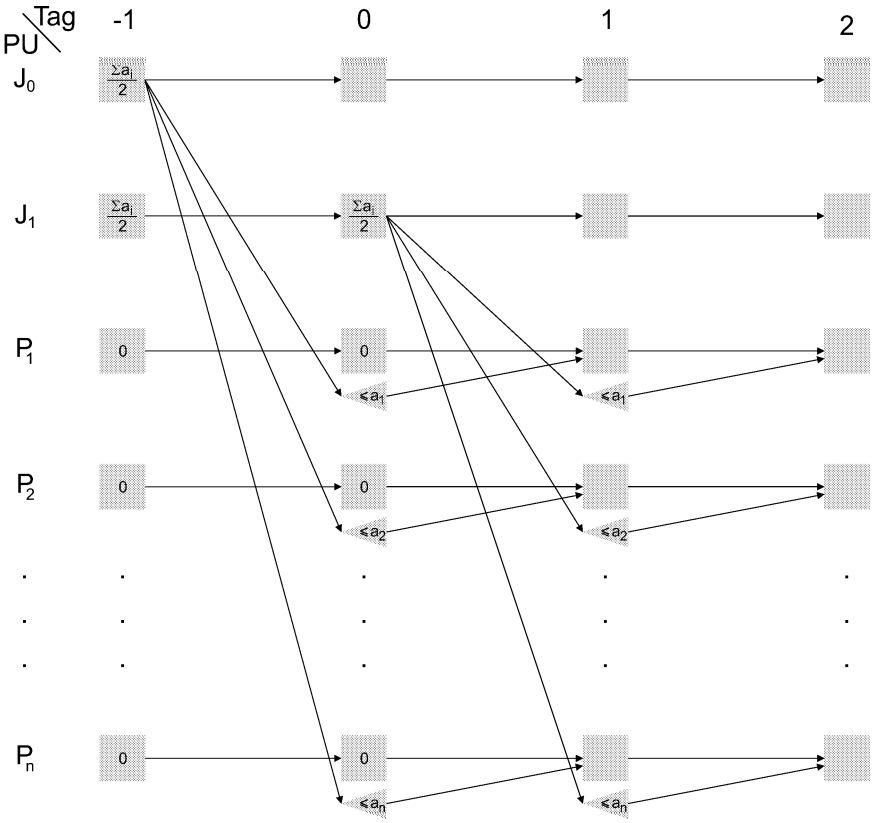


Abbildung 3.1: SPTS_0-Instanz $\phi(n, a)$

Kapitel 4

Lösungsverfahren

Die Anwendung von Optimierungsproblemen zur Einsatzplanung von Arbeitskräften (manpower problems) reicht bis in die 1940er Jahre zurück [9]. Das im Zusammenhang mit Pilotenplanung in der Literatur am häufigsten untersuchte Problem ist das der *Umlaufplanung*, bei dem Crews für die Flüge eines erstellten Flugplans eingeteilt werden, so dass die zulässigen Maximalflugzeiten nicht überschritten sowie die Wartezeiten der Piloten an den Flughäfen minimiert werden und jede Crew nach einem *Umlauf* (Folge der geleisteten Flüge) wieder zu ihrem Ausgangspunkt zurückkommt. Eine Problemstellung, die inhaltliche Ähnlichkeiten mit SPTS aufweist, wird in [9] als *Manpower Planning* (MP) eingeführt. Wie bei SPTS ist es hier das Ziel, innerhalb eines Planungshorizontes Kurse unter bestimmten Bedingungen anzulegen. Der wesentliche Unterschied ist, dass in MP Kurse für eine Menge konkreter Piloten geplant werden. Bei SPTS hingegen haben wir aufgrund der ungewissen Bewerbungsentscheidungen einen anonymen Pilotenfluss modelliert. Entsprechend bezieht sich die in [9] angegebene Lösungsheuristik auf die Einteilung konkreter Piloten und ist für SPTS nicht verwendbar. Weitere Arbeiten zu Optimierungsproblemen aus der Luftfahrtindustrie und ähnlichen Planungsproblemen sind zum Beispiel in [1], [3] und [10] zu finden.

Da SPTS nach Kapitel 3 ein NP-vollständiges Problem ist, können wir nicht erwarten, einen Algorithmus zu finden, der in realistischer (polynomieller) Laufzeit einen optimalen Schulungsplan liefert. Eine optimale Lösung wäre aber aufgrund der unsicheren Eingangsdaten (zum Beispiel der prognostizierten Stammverteilung) ohnehin nur eine Approximation an die Wirklichkeit, weshalb es vollkommen reicht, statt eines optimalen Schulungsplanes einen ausreichend *guten* zu erzeugen. Dabei stellt sich die Frage: Wie können wir einschätzen, wie gut ein Schulungsplan $x \in X$ ist? Würden wir den Optimalwert z des Problems SPTS bereits kennen, könnten wir die prozentuale Abweichung des Zielfunktionswertes cx vom Optimalwert berechnen und wüssten, wie weit wir noch vom Optimalwert entfernt sind. Leider kennen wir den optimalen Zielfunktionswert z nicht, weshalb wir eine andere Referenzgröße für die Güte von Schulungsplänen brauchen. Hier kommt die Idee unterer Schranken ins Spiel: Können wir auf einfache Art und Weise eine untere Schranke $l \leq z$ für den Optimalwert z berechnen, so können wir diese, anstelle des Optimalwertes, als Referenzgröße verwenden. Denn liegt der Zielfunktionswert cx eines Schulungsplanes x näher an der unteren Schranke l als der Zielfunktionswert cy eines Schulungsplanes y, so liegt cx auch näher am optimalen Zielfunktionswert z als cy. Die relative Abweichung eines Schulungsplanes $x \in X$ von einer unteren Schranke l nennen wir **Gap** (Lücke). Wir berechnen den Gap als Quotient aus absoluter Differenz zur Schranke und absolutem Zielfunktionswert:

$$\text{gap}(x, l) := \frac{|cx - l|}{|cx| + 10^{-10}}$$

Der Summand 10^{-10} im Nenner soll verhindern, dass durch 0 geteilt wird, falls der Zielfunktionswert cx eines Schulungsplanes x gleich 0 ist. Die dadurch verursachte Wertänderung des Quotienten können wir problemlos vernachlässigen, da unsere Zielfunktion die Einheit [€] besitzt und daher höchstens auf zwei Dezimalstellen relevant ist. Im folgenden Abschnitt werden wir das Konzept der *linearen Relaxierung* verwenden, um untere Schranken für den Optimalwert z zu berechnen.

4.1 Relaxierung

Für ein gemischt ganzzahliges Problem (MIP) $z = \min\{\ cx\ :\ x \in \mathbb{R}^n \times \mathbb{Z}^m,\ Ax \leq b\ \}$ heiße das Problem (MIP′) $z' = \min\{\ cx\ :\ x \in \mathbb{R}^{n+m},\ Ax \leq b\ \}$ die **lineare Relaxierung** von MIP [8].

Durch das Aufheben der Ganzzahligkeitsbedingungen an die letzten m Variablen des Vektors x sind für diese Variablen nun auch sämtliche fraktionalen Werte zulässig, für die das Gleichungssystem $Ax \leq b$ erfüllt ist. Die Menge X' der zulässigen Lösungen von MIP′ ist damit mindestens so groß wie die Menge X der zulässigen Lösungen von MIP, es gilt $X \subseteq X'$. Da im Problem MIP′ das Minimum über eine größere zulässige Lösungsmenge genommen wird, kann der optimale Zielfunktionswert höchstens kleiner werden, d.h. es gilt $z' \leq z$. Damit ist der optimale Zielfunktionswert der Relaxierung eine untere Schranke für den optimalen Zielfunktionswert des Originalproblems. Obwohl sich die Definitionen von MIP und MIP′ sehr ähnlich sehen, können wir den Optimalwert des Problems MIP′ verhältnismäßig leicht berechnen. Durch das Fallenlassen der Ganzzahligkeitsbedingungen nämlich wird MIP′ zum **linearen Optimierungsproblem**. Diese Problemklasse ist in Literatur und Praxis bereits gut untersucht worden und mit Hilfe polynomieller Verfahren [5] oder des Simplexalgorithmus [6] am Computer einfach zu lösen.

Sehen wir uns nun an, was beim Übergang unseres Problems SPTS zur linearen Relaxierung SPTS′ geschieht. Da sämtliche Ganzzahligkeitsbedingungen fallengelassen werden, ändern sich die Definitionsbereiche der Kursvariablen x^{C}, der Indikatorvariablen x^{I} und der Zerlegungsvariablen $x^{(4)}$, $x^{(2)}$, $x^{(1)}$ jeweils zu $\mathbb{R}$. Dadurch entstehen folgende (unerwünschte) Effekte:

- Durch die Relaxierung der Kursvariablen x^{C} ist es möglich, Kurse mit fraktionalen Traineezahlen anzulegen. Wir werden diese Kurse als **fraktionale Kurse** bezeichnen. Fraktionale Kurse existieren in der Realität nicht, da immer nur ganze Trainees an Kursen teilnehmen.

- Durch die Relaxierung der Indikatorvariablen x^{I} können Kurse nicht mehr nur *angelegt* (=1) oder *nicht angelegt* (=0) werden, sondern auch dazwischen liegende Zustände annehmen; solche Werte der Indikatorvariablen haben allerdings keine Interpretation in der Realität. Die Kursabstands-Nebenbedingung (2.13) ist bei fraktionalen Indikatorvariablen nicht mehr einschränkend genug, weshalb *fraktional-angelegte* Kurse den Kursabstand verletzen können.

- Durch die Relaxierung der Variablen $x^{(4)}$, $x^{(2)}$, $x^{(1)}$ ist die teilbinäre Kursdarstellung nicht mehr eindeutig. Insbesondere sind nicht durch 4 teilbare Kursstärken x^{C} durch alleinige Verwendung der Variable $x^{(4)}$ darstellbar ($x^{(4)} := x^{\mathrm{C}}/4$), wodurch der Sinn der teilbinären Darstellung – die Aufsplittung in Quadrupel, Paare und ungeraden Trainee, um bestimmte anfallende Kosten jeweils auf Paare oder Quadrupel aufzurunden – verloren geht.

In relaxierten Lösungen müssen Kurse nicht mehr zwangsläufig die gegebenen Mindestabstände einhalten und ganzzahlige Kursstärken haben; vielmehr kann es passieren, dass mehrere dicht aufeinander folgende Kurse mit nur Traineebruchteilen angelegt werden. Kurz: Es kommt zu einer Art Verwischungseffekt statt zu klar abgegrenzten ganzzahligen Kursen. Machen wir uns an einem Beispiel klar, wieso verwischte Kurse im relaxierten Problem meist besser als abgegrenzte fixierte Kurse sind: Ist der Bedarf auf einer PU während einer Zeitperiode etwa konstant und gehen nur kontinuierlich einige Piloten durch Fluguntauglichkeit etc. ab, so ist es sicher kostenaufwändiger, alle zwei Monate einen Kurs mit beispielsweise 8 Trainees auf die PU anzulegen und dadurch das Pilotendefizit schubweise auszugleichen, anstatt viele aufeinander folgende Kurse mit jeweils nur 0.5 Trainees anzulegen und so den prozentualen Pilotenabgang ständig zu kompensieren.

Unser eingangs genanntes Ziel, eine untere Schranke l für den Optimalwert z von SPTS zu berechnen, haben wir mit $l := z' \leq z$ erreicht. Damit steht uns die gewünschte Referenzgröße für die Güte von Schulungsplänen zur Verfügung. Durch die Relaxierung haben wir aber noch mehr als eine bloße Schranke gewonnen: Mit der Optimallösung von SPTS′ steht uns ein Schulungsplan aus fraktionalen Kursen zur Verfügung. Von ihm ausgehend werden wir im folgenden Abschnitt durch sukzessives Einschränken der Wertebereiche der fraktionalen Variablen entweder zeigen können, dass gar kein Schulungsplan für SPTS existiert, oder einen Schulungsplan mit Gütegarantie finden.

4.2 Branch & Cut

Der Branch&Cut-Algorithmus konstruiert einen binären Baum, bei dem jeder Knoten ein lineares Optimierungsproblem darstellt. Zu Beginn des Algorithmus legen wir den Wurzelknoten an, der das relaxierte Problem SPTS′ enthält; weitere Knoten hat der Baum noch keine. Nun lösen wir das Wurzelproblem, beispielsweise mit Hilfe des Simplexalgorithmus [6]. Existiert keine Lösung für die Relaxierung, so besitzt SPTS wegen $X \subseteq X'$ ebenfalls keine Lösung und wir beenden den Algorithmus (mit dem Beweis, dass es keinen Schulungsplan gibt). Andernfalls untersuchen wir, ob die letzten m Variablen der relaxierten Optimallösung noch fraktionale Werte enthalten. Sollten all jene Variablen bereits ganzzahlig sein, so ist die relaxierte Lösung wegen $X \subseteq X'$ auch optimale Lösung für SPTS und wir beenden gleichfalls den Algorithmus (mit der Optimallösung). Ist mindestens eine der letzten m Variablen noch fraktional, so versuchen wir, Schnitte zu finden. **Schnitte** (**Cuts**) sind lineare Ungleichungen, die von der zulässigen Lösungsmenge der Relaxierung Gebiete abschneiden, die lediglich fraktionale Lösungen enthalten [8]. Die dahinterstehende Idee ist, die zulässige Lösungsmenge zu verkleinern, ohne ganzzahlige Lösungen zu eliminieren.

- Besteht für drei Binärvariablen x, y, z die Nebenbedingung $30x + 35y + 40z \leq 50$, können wir die zulässige Lösungsmenge durch Hinzufügen des Schnittes $1x + 1y + 1z \leq 1$ verkleinern. Es lässt sich leicht überprüfen, dass dieser Schnitt keine ganzzahligen Lösungen eliminiert, wohl aber einige fraktionale wie beispielsweise $(1.0, 0.2, 0.3)$.

Um die Probleminstanz auf einer vernünftigen Größe zu halten, werden wir lediglich eine begrenzte Anzahl von Schnitten erzeugen. Diese fügen wir zum Wurzelproblem hinzu und lösen es erneut. Entweder tritt nun einer der oben genannten Fälle ein (es existiert kein Schulungsplan oder wir haben einen optimalen Schulungsplan gefunden) oder es gibt immer noch eine fraktionale Variable, sagen wir mit dem Wert $a \notin \mathbb{Z}$. Ein ganzzahliger Wert für diese Variable muss entweder mindestens so groß wie $\lceil a \rceil$ oder höchstens so groß wie $\lfloor a \rfloor$ sein. Wir erzeugen nun zwei neue Knoten als Nachfolger der Wurzel, die jeweils ein neues Unterproblem enthalten. Die Unterprobleme schränken den Wertebereich der fraktionalen Variablen jeweils wie beschrieben nach oben bzw. unten ein und haben damit disjunkte zulässige Lösungsmengen. Nach weiteren Lösungen werden wir in den beiden neuen Zweigen suchen. Das durchgeführte Verzweigen des Baumes nennt man **Branching**.

- Findet das Branching auf einer binären Variable statt, so wird im einen Zweig die obere Schranke der Variable auf 0 gesetzt, im anderen Zweig die untere Schranke auf 1. Eine binäre Variable ist also in beiden Zweigen sogar bereits auf einen ganzzahligen Wert festgelegt.

Damit ist der Wurzelknoten abgearbeitet und wird inaktiv, d.h. er wird im Weiteren nicht mehr untersucht. Alle noch nicht abgearbeiteten Knoten nennen wir **aktiv**. Im Moment sind genau die beiden Nachfolger der Wurzel aktiv. Wir wählen nun nach einem bestimmten Kriterium (siehe 4.2.1) einen der aktiven Knoten aus und verfahren ähnlich wie mit dem Wurzelknoten. Als erstes lösen wir das entsprechende lineare Unterproblem. Ist das Problem unzulässig, enthält der Zweig des aktuellen Knotens keine zulässigen Lösungen und wir löschen den Knoten; im Gegensatz zum Wurzelknoten beenden wir aber den Algorithmus in diesem Falle nicht, sondern wählen einen neuen aktiven Knoten, da in dessen Zweig ja weitere zulässige Lösungen vorhanden sein können. Ist das Problem zulässig, so prüfen wir wieder, ob die optimale Lösung ganzzahlig ist oder nicht. Ist sie ganzzahlig, dann merken wir sie uns als die *beste bisher gefundene ganzzahlige Lösung* (**Incumbent**); hatten wir bereits eine solche gefunden, merken wir uns die neue Lösung nur dann als Incumbent, wenn sie einen besseren (kleineren) Zielfunktionswert hat. Bei jeder neuen Incumbent-Lösung werden vom Baum alle Knoten mit schlechterem Zielfunktionswert entfernt, denn der Zielfunktionswert der Incumbent-Lösung ist eine obere Schranke für den Optimalwert z. Ist die optimale Lösung des aktuellen Knotens nicht ganzzahlig, so überprüfen wir, ob sie einen der bereits erzeugten Schnitte verletzt, und erzeugen zusätzlich neue Schnitte für den aktuellen Zweig des Baumes. Jeden dieser Schnitte fügen wir zum Unterproblem hinzu und optimieren erneut. Wird das Problem dabei zu irgendeinem Zeitpunkt unzulässig, löschen wir den aktuellen Knoten. Ansonsten wählen wir nach einem bestimmten Kriterium (siehe 4.2.2) eine fraktionale Variable und beginnen erneut mit dem Branching. Anschließend wird der aktuelle Knoten inaktiv.

Der Branch&Cut-Algorithmus läuft so lange, bis keine aktiven Knoten mehr vorhanden sind oder ein bestimmtes Limit erreicht wurde (siehe 4.2.3). Zu jedem Zeitpunkt des Algorithmus gibt es einen Knoten, dessen Zielfunktionswert kleiner als der aller anderen Knoten ist. Jener Zielfunktionswert (**Best Node Value**) ist stets die beste bekannte untere Schranke l für den optimalen Zielfunktionswert z von SPTS. Berechnen wir in jeder Iteration des Algorithmus den Gap zwischen l und dem Zielfunktionswert der Incumbent-Lösung, wissen wir stets, wie nahe wir einer optimalen Lösung bereits sind. Läuft der Algorithmus, bis keine aktiven Knoten mehr übrig sind, so wird entweder kein Incumbent gefunden worden sein (womit wir gezeigt haben, dass es überhaupt keinen Schulungsplan gibt), oder der Gap zwischen unterer Schranke l und Incumbent-Zielfunktionswert wird gleich 0 sein (womit die Optimalität der aktuellen Incumbent-Lösung gezeigt ist). Bricht der Algorithmus aufgrund eines Limits vor der Abarbeitung aller aktiven Knoten ab, so ist entweder kein Incumbent gefunden worden (d.h. wir konnten im gegebenen Limit keinen Schulungsplan finden), oder uns steht mit der Incumbent-Lösung ein Schulungsplan x zur Verfügung, für dessen Zielfunktionswert wir mittels $\text{gap}(x, l)$ eine Gütegarantie in Prozent angeben können.

Eine Implementierung des beschriebenen Branch&Cut-Algorithmus findet sich im Programmpacket ILOG CPLEX [2], welches wir zum Lösen von SPTS benutzen. Als Kriterien für die *Auswahl des nächstens Knotens* und die *Auswahl der fraktionalen Variable* stehen verschiedene Optionen zur Verfügung, die nachstehend kurz erläutert werden. Darüber hinaus benennen wir die möglichen *Limits* für einen Abbruch des Algorithmus vor Abarbeitung aller aktiven Knoten.

4.2.1 Auswahl des nächsten Knotens

Die Wahl des aktiven Knotens, der im Branch&Cut-Baum als nächstes untersucht werden soll, kann durch zwei Parameter beeinflusst werden. Der erste Parameter gibt die Wahrscheinlichkeit dafür an, dass vom aktuellen Knoten aus einfach auf einem der beiden angelegten Zweige tiefer in den Baum hinabgestiegen und weitere fraktionale Variablen eingeschränkt werden. Wird die Suche nicht vom aktuellen Knoten aus fortgesetzt, bestimmt der zweite Parameter die Wahl des neuen aktiven Knotens. Die von uns für diesen Parameter verwendete Einstellung ist *Best Bound*, wodurch derjenige aktive Knoten mit dem besten Zielfunktionswert als nächster untersucht wird.

4.2.2 Auswahl der fraktionalen Variable

Die Wahl der fraktionalen Variable, auf der das Branching durchgeführt werden soll, kann durch einen Parameter beeinflusst werden. Die Einstellung *Min Infeasbile* wählt eine fraktionale Variable, deren Wert einer ganzen Zahl am nächsten kommt. Entsprechend wählt *Max Infeasbile* eine fraktionale Variable, deren Wert am weitesten von einem ganzzahligen entfernt ist. Die Wahl der Branching-Variable kann mittels *Default* auch CPLEX selbst überlassen werden. Darüber hinaus besteht die Möglichkeit, eine Rangfolge (*Priority Order*) innerhalb der Variablen festzulegen, die die Branching-Variable stets eindeutig bestimmt. Für die Entscheidung, welcher der beiden Zweige eines Branchings zuerst untersucht werden soll, steht ein weiterer Parameter mit den Einstellungen *Up*, *Down* (Richtung des Branchings) und *Default* (CPLEX trifft die Entscheidung) zur Verfügung.

4.2.3 Limits

Durch das Setzen von Limits kann ein Abbruch des Branch&Cut-Algorithmus erreicht werden, bevor alle aktiven Knoten abgearbeitet sind. Limits können für die *Rechenzeit*, die *Anzahl der Knoten*, die *Baumgröße im Speicher* und die bereits gefundene *Anzahl ganzzahliger Lösungen* festgelegt werden. Außerdem ist es möglich, einen bestimmten Gap vorzugeben (z.B. 1%), bei dessen Erreichen der Algorithmus beendet wird. Mit der zu diesem Zeitpunkt verfügbaren Incumbent-Lösung haben wir dann einen Schulungsplan, der um höchstens den vorgegebenen Gap von einem optimalen Schulungsplan abweicht.

4.3 Heuristik

Mit dem Branch&Cut-Algorithmus können wir Schulungspläne von beliebig klein vorgebbarem Gap erzeugen, wir haben also eine **a priori-Gütegarantie** (Gütegarantie im Vorhinein) für den erstellten Schulungsplan. Der Nachteil besteht darin, dass es unter Umständen sehr lange dauern kann, bis im Branch&Cut-Baum ein Schulungsplan von gewünschtem Gap gefunden worden ist. In diesem Abschnitt werden wir deshalb einen heuristischen Algorithmus entwickeln, der in vielen Fällen schneller gute Schulungspläne erzeugt, für die es allerdings keine a priori-Gütegarantie gibt. Die Zielfunktionswerte des heuristischen bzw. des relaxierten Schulungsplanes liefern aber eine obere bzw. untere Schranke für den Optimalwert z, womit wir eine **a posteriori-Gütegarantie** (Gütegarantie im Nachhinein) für den heuristischen Schulungsplan angeben können.

Wir starten wie bei Branch&Cut mit der Lösung des relaxierten Problems SPTS′ und nutzen die Tatsache aus, dass uns der in 4.1 beschriebene Verwischungseffekt von Kursen bereits die grobe Richtung zeigt, auf welchen PUs und in welchen Zeiträumen Kurse angelegt werden sollten: Summieren wir alle fraktionalen Trainees einer PU in einem bestimmten Zeitraum auf, wissen wir ungefähr, wie viele Trainees in einem guten Schulungsplan während dieses Zeitraums auf der PU benötigt werden. Wir ersetzen nun die fraktionalen Kurse durch *einen* ganzzahligen Kurs innerhalb des Zeitraums und schulen die berechnete Traineesumme komplett mit diesem Kurs um. Weil wir noch nicht wissen, bei welchem Knoten des Zeitraums es am sinnvollsten wäre, den Summenkurs anzulegen, legen wir ihn zunächst auf den ersten Knoten des Zeitraums und versuchen später, ihn noch zu verschieben, um den Schulungsplan zu verbessern. Unsere Heuristik besteht damit aus zwei Teilen, nämlich einer *Startheuristik*, die zunächst einen in vielen Fällen guten Schulungsplan erstellt, und einer *Augmentationsheuristik*, die ihn noch zu verbessern versucht. In den Pseudocodes der Algorithmen werden wir die beiden folgenden Funktionen als Interface zu CPLEX verwenden:

- *Optimize* ruft den Lösungsalgorithmus von CPLEX für das aktuelle lineare Optimierungsproblem auf und besitzt die Rückgabeparameter $(state, x)$. Hat *state* den Wert *infeasible*, so existiert keine zulässige Lösung. Andernfalls existiert aufgrund der Beschränktheit der Zielfunktion ($z \geq 0$) sogar immer eine optimale Lösung, die in x zurückgegeben wird.
- *Change lower and upper bounds* setzt für eine Variable die obere und untere Schranke auf denselben angegebenen Wert, d.h. fixiert die Variable in CPLEX. Existieren bereits Nebenbedingungen für die Schranken einer Variable, brauchen also nur deren rechte Seiten geändert zu werden. Existieren noch keine Schranken, werden einfach neue Nebenbedingungen erstellt.

Wenn wir die teilbinären Kursvariablen $x^{(4)}$, $x^{(2)}$, $x^{(1)}$ auf ganze Zahlen fixieren, dann werden auch die Kursvariablen x^{C} ganzzahlig. Um der Tatsache Rechnung zu tragen, dass im Abstandsbereich jedes Knotens nur höchstens ein Kurs angelegt werden darf (2.13), wählen wir als Zeiträume, über die wir die fraktionalen Trainees summieren, jeweils gerade die Abstandsbereiche $D_{p,d}^{\mathrm{post}}$ (Seite 18) der Kurse und haben damit auch die Indikatorvariablen x^{I} im Griff. Wir partitionieren für die Heuristik die Diskretisierung jeder PU p in 5 Teile, die an jeder Stelle global zugreifbar sind:

$$D_p = D_p^{\mathrm{lock}} \dot\cup D_p^{\mathrm{frac}} \dot\cup D_p^{\mathrm{fix}} \dot\cup D_p^{\mathrm{post}} \dot\cup D_p^{\mathrm{zero}}$$

Jedes Intervall gehört zu jedem Zeitpunkt des Algorithmus zu genau einer Partition. Mittels der Partitionen merken wir uns die Fixierungszustände der Kurse. Die erste Partition D_p^{lock} soll genau die Intervalle enthalten, deren Kursvariablen von Vornherein durch Nebenbedingungen auf einen festen Wert gesetzt sind. Sie können von der Heuristik nicht mehr verändert werden und heißen deshalb *gesperrte* Intervalle. Wir setzen diese Partition im Algorithmus auf $D_p^{\text{near freez}}$ (Seite 19), da jene Kurse bereits durch die Nebenbedingungen (2.7) und (2.14) bestimmt sind. Wollen wir zusätzlich beispielsweise alle Kurse sperren, die erst nach dem Strategiezeitraum enden und daher zwar Trainees von den Stamm-PUs abziehen, sie aber nicht mehr der Ziel-PU zuführen, müssen wir bloß die entsprechenden Variablen durch Nebenbedingungen auf 0 setzen und die Intervalle in D_p^{lock} aufnehmen. Alle nicht gesperrten Intervalle ordnen wir zu Beginn des Algorithmus der Partition D_p^{frac} zu, die so zu verstehen ist, dass hier noch fraktionale Kurse angelegt werden dürfen.

Fixieren wir während der Heuristik einen Kurs auf eine natürliche Traineezahl, dann verschieben wir das zugehörige Intervall in die Partition D_p^{fix}. Das Verschieben eines einzelnen Intervalls d oder analog einer Intervallmenge M in eine andere Partition kennzeichnen wir durch die Schreibweise $D_p^{\cdots} \curvearrowleft d$ bzw. $D_p^{\cdots} \curvearrowleft M$. Sobald ein Kurs $x_{p,d}^{\text{C}}$ fixiert wurde, dürfen im Abstandsbereich $D_{p,d}^{\text{post}}$ des Kurses keine weiteren Schulungen mehr angelegt werden. In der Partition D_p^{post} werden wir deshalb jeweils genau die Intervalle speichern, die *im Abstandsbereich eines fixierten Kurses liegen.* Einen Kurs werden wir nur dann fixieren, wenn er zum einen nicht in D_p^{post} liegt, und wenn zum anderen auch *sein Abstandsbereich keinen fixierten Kurs enthält*, also wenn $D_{p,d}^{\text{post}} \cap D_p^{\text{fix}} = \varnothing$ gilt. Schließlich verbleiben noch die Intervalle, die nicht im Abstandsbereich eines fixierten Kurses liegen, bei denen wir aber auch keinen Kurs anlegen, sondern die Kursvariable explizit auf den Wert 0 setzen wollen. Jene Intervalle speichern wir in der Partition D_p^{zero}.

Die folgende Funktion *Set Course* werden wir in Start- und Augmentationsheuristik verwenden, um die Teilnehmerzahl eines Kurses auf einen ganzzahligen Wert zu setzen bzw. auf einen anderen ganzzahligen Wert zu ändern. Die Zeilen 1–4 berechnen die teilbinäre Zerlegung der Traineezahl s und fixieren die entsprechenden Variablen in CPLEX. In den restlichen Zeilen verschieben wir das zugehörige Intervall d und seinen Abstandsbereich $D_{p,d}^{\text{post}}$ in die passenden Partitionen.

function *Set Course*(s, p, d)

1 *Change lower and upper bound of variable* $x_{p,d}^{(1)}$ *to* $s \bmod 2$

2 *Change lower and upper bound of variable* $x_{p,d}^{(2)}$ *to* $\begin{cases} 1 & \textbf{if } s \bmod 4 \geq 2 \\ 0 & \textbf{else} \end{cases}$

3 *Change lower and upper bound of variable* $x_{p,d}^{(4)}$ *to* $s \operatorname{div} 4$

4 *Change lower and upper bound of variable* $x_{p,d}^{I}$ *to* $\begin{cases} 1 & \textbf{if } s > 0 \\ 0 & \textbf{else} \end{cases}$

5 **if** $s = 0$

6 **if** $d \in D_p^{\text{fix}}$

7 $D_p^{\text{zero}} \curvearrowleft D_{p,d}^{\text{post}}$

8 **else if** $d \notin D_p^{\text{post}}$

9 $D_p^{\text{zero}} \curvearrowleft d$

10 **else if** $d \in D_p^{\text{frac}} \cup D_p^{\text{zero}}$

11 $D_p^{\text{fix}} \curvearrowleft d$

12 $D_p^{\text{post}} \curvearrowleft D_{p,d}^{\text{post}} \setminus \{d\}$

4.3.1 Startheuristik

Die Startheuristik verläuft folgendermaßen: Wir lösen mit Hilfe von CPLEX das relaxierte Problem SPTS′ und ermitteln danach mit der Funktion *Calculate Maximal Post Sum* den Kurs, in dessen Abstandsbereich die Summe s der fraktionalen Trainees am größten ist. Anschließend fixieren wir den Kurs auf die nächstgrößere ganze Zahl $\lceil s \rceil$ bzw. auf die maximal zulässige Teilnehmerzahl $\text{cmax}(p, \overline{d})$, falls diese kleiner als $\lceil s \rceil$ ist. Wird das Problem beim Fixieren des Kurses unzulässig, setzen wir die Traineezahl auf einen niedrigeren Wert und optimieren erneut. Im Algorithmus geschieht dies mit der einfachen Dekrementfunktion $s \mapsto s - 1$. Alternativ ist an dieser Stelle eine beliebige streng monoton fallende Funktion $\mathbb{N} \to \mathbb{N}_0$ einsetzbar. Nachdem der Kurs zulässig fixiert wurde, bilden wir erneut die Traineesummen über die Abstandsbereiche der Knoten und wählen den nächsten zu fixierenden Kurs aus. Dies wird wiederholt, bis die maximale Traineesumme einen bestimmten Schwellwert (im Algorithmus $\frac{1}{2}$) unterschreitet oder der Algorithmus aufgrund einer im Rahmen der Heuristik nicht behebbaren Problem-Unzulässigkeit abgebrochen wird (abort). Der jeweils zu fixierende Kurs kann statt über die maximale Traineesumme des Abstandsbereichs alternativ als der Kurs gewählt werden, bei dem das Verhältnis jener Traineesumme zur maximalen Teilnehmerzahl des Kurses am größten ist. Die Funktion *Annul Remaining Courses* setzt schließlich alle verbleibenden Kurse auf den Wert 0. Damit konstruiert die Startheuristik de facto einen ganzzahligen Schulungsplan x.

function *Start Heuristic*

1 **for each** $p \in PU$
2 $\quad D_p^{\text{lock}} \leftarrow D_p^{\text{near freez}}$
3 $\quad D_p^{\text{frac}} \leftarrow D_p \setminus D_p^{\text{lock}}$
4 $(state, x) \leftarrow$ *Optimize*
5 **if** $state =$ *infeasible*
6 $\quad$ **abort**
7 $(s, p, d) \leftarrow$ *Calculate Maximal Post Sum*(x)
8 **if** $s \geq \frac{1}{2}$
9 $\quad s \leftarrow \min\{\lceil s \rceil, \text{cmax}(p, \overline{d})\}$
10 $\quad$ *Set Course*(s, p, d)
11 $\quad (state, x) \leftarrow$ *Optimize*
12 $\quad$ **if** $state =$ *infeasible*
13 $\quad\quad$ **if** $s = 0$
14 $\quad\quad\quad$ **abort**
15 $\quad\quad s \leftarrow s - 1$
16 $\quad\quad$ **goto** 10
17 $\quad$ **goto** 7
18 *Annul Remaining Courses*
19 $(state, x) \leftarrow$ *Optimize*
20 **if** $state =$ *infeasible*
21 $\quad$ **abort**
22 **return** x

function *Calculate Maximal Post Sum*(x)

1 $s' \leftarrow 0$
2 **for each** $(p, d) \in K$
3 $\quad$ **if** $\left(d \in D_p^{\text{frac}}\right) \wedge \left(D_{p,d}^{\text{post}} \cap D_p^{\text{fix}} = \varnothing\right)$
4 $\quad\quad s \leftarrow \sum_{e \in D_{p,d}^{\text{post}}} x_{p,e}^{\text{C}}$
5 $\quad\quad$ **if** $s > s'$
6 $\quad\quad\quad (s', p', d') \leftarrow (s, p, d)$
7 **return**[1] (s', p', d')

function *Annul Remaining Courses*

1 **for each** $(p, d) \in K$
2 $\quad$ **if** $d \in D_p^{\text{frac}} \cup D_p^{\text{post}}$
3 $\quad\quad$ *Set Course*$(0, p, d)$

[1] Gilt $s' = 0$ beim Aufruf von Return, so haben p' und d' keinen definierten Wert. Dies ist aber unproblematisch, da die Rückgabewerte beider Parameter in diesem Falle von der Startheuristik gar nicht verwendet werden.

4.3.2 Augmentationsheuristik

Nach erfolgreicher Startheuristik ist D_p^{frac} leer und die Diskretisierung jeder PU besteht nur noch aus 4 Partitionen. Mit der Augmentationsheuristik werden wir nun, ausgehend vom übergebenen ganzzahligen Schulungsplan x der Startheuristik, eine Folge ganzzahliger Schulungspläne erzeugen, deren Zielfunktionswerte jeweils besser werden (sinken). Mittels unten stehender Funktion *Modify Course* werden wir in der Augmentationsheuristik versuchsweise lokale Veränderungen am jeweils aktuellen Schulungsplan vornehmen. Als Parameter übergeben wir den aktuellen Schulungsplan x, einen Knoten $(p, d) \in K$, bei dem ein fixierter Kurs angelegt ist, und eine Methode m, mit der die Funktion den fixierten Kurs modifizieren soll. Für $m \in \{dec, inc\}$ verkleinern bzw. vergrößern wir die Traineezahl des Kurses, sofern sie dabei weiterhin zulässig bleibt, und für $m \in \{pre, next\}$ verschieben wir den Startzeitpunkt auf das vorherige bzw. nachfolgende Diskretisierungsintervall, falls ein solches existiert, sich dort noch kein anderer fixierter Kurs befindet, kein Abstandsbereich verletzt wird und die Traineezahl $x_{p,d}^{\text{C}}$ auch für die Kursvariable des neuen Intervalls zulässig ist.

function *Modify Course*(x, p, d, m)

```
1   case m of
2     dec  :  if x^C_{p,d} − 1 ≥ 0
3                 Set Course(x^C_{p,d} − 1, p, d)
4     inc  :  if x^C_{p,d} + 1 ≤ cmax(p, d̄)
5                 Set Course(x^C_{p,d} + 1, p, d)
6     pre  :  if (∃ pre_p d ∈ D_p^zero) ∧ (D^post_{p,pre_p d} ∩ D_p^fix \ {d} = ∅) ∧ (x^C_{p,d} ≤ cmax(p, pre_p d))
7                 Set Course(0, p, d)
8                 Set Course(x^C_{p,d}, p, pre_p d)
9     next :  if (∃ next_p d ∈ D_p^zero ∪ D_p^post) ∧ (D^post_{p,next_p d} ∩ D_p^fix = ∅) ∧ (x^C_{p,d} ≤ cmax(p, next_p d))
10                Set Course(0, p, d)
11                Set Course(x^C_{p,d}, p, next_p d)
```

Die Funktion *Augmentation* wendet nun nacheinander alle Methoden aus *Modify Course* auf einen fixierten Kurs (p, d) des aktuellen Schulungsplans x an, berechnet den neuen Zielfunktionswert und macht die Änderungen mittels *Modify Course* wieder rückgängig, wobei $opp(m)$ die jeweils andere Operation der Paare $inc \leftrightarrow dec$ bzw. $pre \leftrightarrow next$ bezeichnet. Wird mit einer Methode m ein Erfolg erzielt, machen wir den modifizierten zum aktuellen Schulungsplan und wenden danach auf den fixierten Kurs erneut die Methoden aus *Modify Course* an. Sinnvollerweise schließen wir dabei die Methode $opp(m)$ aus, da sie wieder den vorherigen Schulungsplan liefern würde. Dieser Vorgang wird so lang wiederholt, bis keine Verbesserung mehr erzielt oder die Kursstärke auf 0 gesetzt wird.

Die Augmentationsheuristik schließlich beginnt beim ersten fixierten Kurs einer festen PU p und wendet auf ihn die Funktion *Augmentation* an. Wenn keine Verbesserung erzielt werden konnte, geht sie zum nächsten fixierten Kurs der PU über, ansonsten wird das zugehörige Intervall d als Continue-Intervall c gespeichert und dann der vorangegangene Kurs (falls ein solcher existiert) erneut untersucht, bevor vom Continue-Intervall c aus weitergesucht wird. Die Funktion *Optimize* wird damit höchstens quadratisch oft aufgerufen, sofern die maximale Kursstärke cmax durch eine Konstante nach oben beschränkt ist. Benutzt *Optimize* einen polynomiellen Algorithmus [5] zum Lösen des aktuellen linearen Unterproblems, dann hat die Augmentationsheuristik insgesamt polynomielle Laufzeit. Eine Variante der Heuristik entsteht durch das Entfernen der Zeilen 4, 10 und 12, wodurch potentiell bessere Lösungen gefunden werden können, da kein Continue-Knoten gespeichert und damit ein größerer Teil des zulässigen Lösungsraumes abgesucht wird. Allerdings kann das Verfahren ohne diese Zeilen eine exponentielle Laufzeit bekommen. Es sei noch bemerkt, dass die Durchmusterungsreihenfolge der Planungseinheiten im Algorithmus beliebig gewählt wird und die entstehende Folge augmentierender Schulungspläne damit von jener Reihenfolge abhängt.

function *Augmentation Heuristic*(x)

```
1   for each p ∈ PU
2       if Exists a fixed course on PU p
3           b ← Interval of the first fixed course on PU p
4           c = b
5           d = b
6           (augmented, x, d) ← Augmentation(x, p, d)
7           if augmented = true
8               if Exists a fixed course before interval d
9                   b ← Interval of the latest fixed course before interval d
10                  c = max{c, d}
11                  goto 5
12          d = max{c, d}
13          if Exists a fixed course after interval d
14              b ← Interval of the earliest fixed course after interval d
15              goto 5
16  return x
```

function *Augmentation*(x, p, d)

```
1   augmented ← false
2   M ← {dec,inc,pre,next}
3   M^opp ← ∅
4   for each m ∈ M \ M^opp
5       Modify Course(x, p, d, m)
6       (state, x) ← Optimize
7       Modify Course(x, p, d, opp(m))
8   if At least one method augmented the objective value
9       augmented ← true
10      m ← A method of M \ M^opp with maximal augmentation
11      Modify Course(x, p, d, m)
12      (state, x) ← Optimize
13      if m = pre
14          d = pre_p d
15      if m = next
16          d = next_p d
17      if x^C_{p,d} ≠ 0
18          M^opp ← {opp(m)}
19          goto 4
20  return (augmented, x, d)
```

4.4 Numerische Ergebnisse

Nachdem wir nun zwei verschiedene Algorithmen zur Lösung von SPTS entwickelt haben, werden wir in diesem Abschnitt abschließend einige numerische Ergebnisse für konkrete Probleminstanzen diskutieren. Als Ausgangsmaterial stand hierfür ein kompletter Satz an realen Eingangsdaten für einen Strategiezeitraum von 1900 Tagen zur Verfügung. Die von diesen Daten erzeugte Probleminstanz von SPTS nennen wir im Folgenden die *Originalinstanz*. Um die Algorithmen für mehrere verschiedene Instanzen testen zu können, wurden aus der Originalinstanz durch Variation von ausgewählten Eingangswerten weitere Instanzen erstellt. Das Hauptaugenmerk lag dabei auf der Erzeugung verschiedener Bedarfsszenarien, da diese auch die wesentlichen Einflüsse in der Realität darstellen. Die Algorithmen wurden für folgende Variationen der Originalinstanz getestet:

- **Variation UD** (Uniform Distribution). Für jede PU wird zunächst das Minimum m und das Maximum M des Flugprogrammbedarfs über alle Tage des Strategiezeitraumes ermittelt. Neue Instanzen werden nun erstellt, indem alle Flugprogrammbedarfe im Strategiezeitraum durch eine gleichmäßige Verteilung auf dem Intervall $[m, M]$ mit zufälligen Werten neu belegt werden. Dies geschieht unabhängig voneinander für Einsatzstunden- und Einsatztagewerte. Trotz der *zufällig* erzeugten neuen Bedarfe können wir die variierten Instanzen als realistisch ansehen, da auch die Bedarfe der Originalinstanz bereits starken Schwankungen unterliegen.
- **Variation BB** (Both Bounds). Diese Instanzen werden analog zu den vorherigen erzeugt, nur werden die variierten Bedarfe hier gleichverteilt aus dem an beiden Grenzen um 20% verbreiterten Intervall $[\frac{4}{5}m, \frac{6}{5}M]$ gewählt.
- **Variation UB** (Upper Bound). Wie oben entstehen diese Instanzen durch einen zufälligen gleichverteilten Bedarf, der hier allerdings nur aus dem um 40% nach oben vergrößerten Intervall $[m, \frac{7}{5}M]$ stammt.
- **Variation SI** (Start Increase). Hier übernehmen wir alle Bedarfe der Originalinstanz, aber erlauben durch eine auf dem Intervall $[\frac{4}{5}\text{inc}(p, \underline{T}), \frac{6}{5}\text{inc}(p, \underline{T})]$ gleichverteilte Zufallsfunktion eine Variation des Anfangspilotenbestandes jeder PU p um bis zu 20% nach oben und unten.

Für jede Variation wurden exemplarisch fünf zufällige Instanzen erzeugt und jeweils mit beiden Algorithmen gelöst. Als Diskretisierungen wurden auf allen PUs 5-Tages-Intervalle zugrundegelegt, ausgenommen die eintägigen Intervalle der eingefrorenen Kurse (Abschnitt 2.4.1) und die Anfangs- und Endintervalle der PUs (Abschnitt 2.2.2). Tabelle 4.1 zeigt die Ergebnisse der Algorithmen, durchgeführt auf einem Rechner mit P4-Prozessor (2.6 GHz), 1 GB RAM und 80 GB Festplatte. In der erste Spalte stehen die verschiedenen Probleminstanzen, in den nachfolgenden Spalten sind jeweils die wichtigsten Kenngrößen des Branch&Cut-Algorithmus und der Heuristik dargestellt. Die Spalte Gap^{F} gibt den Gap der ersten im Branch&Cut-Baum gefundenen ganzzahligen Lösung in [%] an (First Gap). Als Limit zum Abbruch des Algorithmus wurde ein Gap von 1% vorgegeben; die Spalte Gap^{B} enthält den Gap des ersten Schulungsplanes, der diesen Prozentsatz unterschreitet (Branch&Cut Gap). Als Kriterium für die Auswahl einer fraktionalen Variable wurde eine *Priority Order* innerhalb der Variablen festgelegt (Abschnitt 4.2.2), die Kursvariablen gegenüber anderen Variablen bevorzugt und unter zwei Kursvariablen stets jene priorisiert, deren Starttermin auf den späteren Zeitpunkt des Strategiezeitraumes fällt. Die Entscheidung über die Branchingrichtung wurde mittels der Einstellung *Default* CPLEX überlassen. In den nächsten beiden Spalten der Tabelle sind die Anzahl der am Ende des Algorithmus im Branch&Cut-Baum befindlichen Knoten und der insgesamt erzeugten Cuts angegeben. Zusätzlich wurde auf den erhaltenen Schulungsplan noch die Augmentationsheuristik angewandt, um einen Vergleich ihrer Wirkungsweise im Kontext des Branch&Cut-Algorithmus zum Kontext des rein heuristischen Algorithmus machen zu können. Die beiden folgenden Spalten Gap^{A} und Augs geben den Gap des augmentierten Schulungsplanes (Augmentation Gap) und die Anzahl der erfolgreichen Augmentationen an. Die Spalte CPU gibt schließlich die Gesamtrechenzeit des Branch&Cut-Algorithmus in [min] an. In den restlichen vier Spalten der Tabelle sind die wichtigsten Kenngrößen der Heuristik zu finden: Gap^{S} gibt den Gap des von der Startheuristik konstruierten Schulungsplanes an, Gap^{A} den Gap des augmentierten Schulungsplanes, Aug die Anzahl der durchgeführten Augmentationen und CPU die Laufzeit der gesamten Heuristik in [min]. Problemspezifischere Ergebnisse wie Zielfunktionswerte und Deltas oder gar erstellte Schulungspläne können hier aus Datenschutzgründen nicht angegeben werden.

	Branch&Cut-Algorithmus							Heuristik			
Prob	Gap^F	Gap^B	Knoten	Cuts	Gap^A	Augs	CPU	Gap^S	Gap^A	Augs	CPU
Orig	58.96	0.67	5161	2	0.65	3	56	1.53	1.34	8	16
UD1	54.13	0.64	830	1	0.55	2	11	0.92	0.86	5	10
UD2	54.43	0.61	1380	27	0.57	4	14	1.29	1.17	15	9
UD3	55.17	0.7	1600	1	0.64	4	15	1.85	1.51	19	12
UD4	55.56	0.55	1161	30	0.50	4	12	1.38	1.27	12	9
UD5	54.30	0.77	1529	1	0.73	5	14	1.66	1.55	11	9
BB1	41.07	0.32	1216	13	0.32	3	13	0.89	0.74	28	9
BB2	43.89	0.41	1200	17	0.36	9	12	0.74	0.64	19	6
BB3	43.04	0.40	1645	3	0.37	5	14	0.89	0.79	16	8
BB4	43.32	0.29	1168	2	0.29	2	11	0.94	0.87	17	7
BB5	43.62	0.32	1538	11	0.32	1	9	1.09	0.96	22	6
UB1	41.43	0.47	1432	39	0.44	8	14	0.80	0.77	5	36
UB2	42.76	0.35	1380	28	0.34	2	15	0.55	0.47	21	12
UB3	42.10	0.32	1484	0	0.29	11	16	0.65	0.62	9	18
UB4	40.89	0.38	976	5	0.36	6	17	0.62	0.57	24	38
UB5	41.18	0.43	1073	3	0.41	9	16	0.88	0.84	17	37
SI1	0.25	0.25	1310	23	0.20	14	15	0.83	0.78	24	12
SI2	41.29	0.52	1242	74	0.51	3	19	0.72	0.65	15	36
SI3	40.33	0.89	1650	14	0.82	24	21	1.01	0.92	17	20
SI4	39.10	0.53	1412	12	0.51	10	20	0.77	0.75	7	25
SI5	39.63	0.52	1390	15	0.47	11	18	0.94	0.88	11	17

Tabelle 4.1: Numerische Ergebnisse

Vergleicht man in Tabelle 4.1 die Laufzeiten der beiden Algorithmen, so lässt sich feststellen, dass die Heuristik in vielen Fällen etwas schneller als Branch&Cut ist. Extremwerte der Laufzeiten gibt es bei Branch&Cut für die Originalinstanz, bei der Heuristik für einige UB- und SI-Instanzen. Erstgenanntes Phänomen untermauert die in Abschnitt 4.3 eingangs getroffene Aussage, dass es bei Branch&Cut manchmal sehr lange dauern kann, bis ein Schulungsplan von gewünschtem Gap gefunden worden ist. In diesem konkreten Fall wurde nach gleicher Rechenzeit wie bei den anderen Instanzen zwar ein guter Schulungsplan gefunden, dieser wies aber nur einen Gap von 1.01% auf. Zum Auffinden eines Schulungsplanes mit einem Gap unter 1% wurden schließlich abermals große Zeitressourcen benötigt und eine bedeutende Anzahl neuer Knoten im Branch&Cut-Baum erzeugt. Wie man an der Instanz SI1 sehen kann, ist es aber durchaus auch möglich, dass bereits der erste vom Branch&Cut-Algorithmus gefundene Schulungsplan das Limit von 1% unterschreitet. Das Phänomen extremer Laufzeiten der Heuristik auf einigen UB- und SI-Instanzen ist hingegen durch die Instanzen selbst zu erklären. Durch die zufällige Datenerzeugung entstanden bei den betreffenden Instanzen auf einigen PUs sehr kleine Anfangspilotenbestände bzw. Zeiträume sehr hoher Bedarfe. Als Ausgleich liefert die relaxierte Lösung in jenen Zeiträumen viele fraktionale Kurse, die die Schulungsressourcen (Checker, Simulatoren etc.) voll ausnutzen. Die Startheuristik fixiert nun Kurse mit der maximalen Traineesumme der Abstandsbereiche. Dadurch reichen die Schulungsressourcen nicht mehr aus und das Problem wird unzulässig (Zeile 12 der Startheuristik). In diesem Fall wird so lang die Kursstärke verkleinert und jeweils erneut optimiert, bis das Problem wieder zulässig ist. Diese Vorgehensweise mindert zwar die Qualität des letztendlich erzeugten Schulungsplanes nicht, führt aber zu einer starken Laufzeiterhöhung des heuristischen Algorithmus. Die aus Tabelle 4.1 berechenbaren durchschnittlichen Rechenzeiten sind für beide Algorithmen mit 16.76 [min] bemerkenswerterweise exakt gleich groß. Beim Vergleich der Anzahlen durchgeführter Augmentationen lässt sich feststellen, dass für den in der Startheuristik erstellten Schulungsplan meist mehr Augmentationen als für den von Branch&Cut erzeugten gefunden werden. Dies ist dadurch erklärbar, dass letztgenannter Schulungsplan in allen getesteten Instanzen bereits einen besseren Gap als erstgenannter hat und es deshalb allgemein schwieriger ist, ihn noch zu verbessern.

Kapitel 5

Virtuelle Kurse

5.1 Problemstellung

In diesem Kapitel werden wir die in Abschnitt 1.2.1 unter Punkt (3) angesprochene prognostische Umwandlung von fixierten zu virtuellen Kursen durchführen. SPTS liefert als Lösung eine Menge fixierter Kurse, also Kurse mit ganzzahligen Teilnehmerzahlen, bei denen die Trainees fraktional entsprechend der Stammverteilung von ihren Stamm-PUs abgezogen werden. Um virtuelle Kurse zu erhalten, müssen wir auch die einzelnen Übergänge, die zu einem angelegten Kurs gehören, ganzzahlig machen. Die Veränderungen an den Schulungen sollen dabei so gering wie möglich sein:

(1) Jeder fixierte Kurs soll in einen virtuellen Kurs gleicher Teilnehmerzahl umgewandelt werden.

(2) Ist die Anzahl $t_{q,p,d} := \mathrm{W}(q, p, \overline{d}\,) \cdot x^{\mathrm{C}}_{p,d}$ der Trainees von PU q, die am fixierten Kurs $x^{\mathrm{C}}_{p,d}$ teilnehmen, schon ganzzahlig, soll sie im umgewandelten virtuellen Kurs beibehalten werden; ist die Anzahl fraktional, so soll sie im virtuellen Kurs höchstens auf die nächstgrößere ganze Zahl $\lceil t_{q,p,d} \rceil$ aufgerundet oder auf die nächstkleinere ganze Zahl $\lfloor t_{q,p,d} \rfloor$ abgerundet werden.

(3) Ist die Anzahl $s_{q,p} := \sum_{d \in D^{\mathrm{fix}}_p} t_{q,p,d}$ aller Trainees von PU q, die innerhalb des Strategiezeitraumes durch fixierte Kurse auf PU p schulen, bereits ganzzahlig, soll dieselbe Anzahl bei Summation über die umgewandelten virtuellen Kurse auftreten; ist die Anzahl fraktional, soll Summation über die virtuellen Kurse höchstens die nächstgrößere ganze Zahl $\lceil s_{q,p} \rceil$ oder die nächstkleinere ganze Zahl $\lfloor s_{q,p} \rfloor$ ergeben.

Die erste Forderung resultiert daraus, dass die Teilnehmerzahlen fixierter Kurse bereits ganzzahlig sind und wir sie deshalb sofort übernehmen können. Durch die zweite Forderung wird garantiert, dass die Zusammensetzungen der virtuellen Kurse so wenig wie möglich von denen der fixierten Kurse abweichen: Die Anzahl der teilnehmenden Trainees jeder Stamm-PU wird höchstens auf die nächstgrößere oder nächstkleinere ganze Zahl gerundet. Nun könnte es noch passieren, dass bei der Umwandlung in virtuelle Kurse die fraktionalen Trainees einer gewissen Stamm-PU immer abgerundet werden, während die fraktionalen Trainees einer anderen Stamm-PU ständig aufgerundet werden. Mit der dritten Forderung sichern wir deshalb, dass sich trotz der Rundung jedes einzelnen Kurses die Gesamtzahlen der zwischen den PUs umschulenden Trainees höchstens auf die nächstgrößere oder nächstkleinere ganze Zahl ändern.

Da sich Kurse auf verschiedene PUs gegenseitig nicht beeinflussen, werden wir uns im Folgenden auf eine feste PU p beschränken und die Ergebnisse dann für alle anderen PUs analog anwenden. Unser Ziel wird es sein, unter Beachtung der drei oben genannten Forderungen alle fixierten Kurse auf PU p in virtuelle Kurse umzuwandeln und die Summe der dabei auftretenden Rundungsfehler zu minimieren. Es wird sich zeigen, dass sich diese Aufgabe als **kostenminimales Flussproblem** (Min Cost Flow Problem) formulieren lässt [4] und optimal lösbar ist.

5.2 Modellierung

Ein **Netzwerk** $N = (V, A)$ bestehe aus einer endlichen Menge V von Knoten und einer Menge $A \subseteq \{(v, w) : v, w \in V, v \neq w\}$ von gerichteten Kanten zwischen diesen Knoten. Wir konstruieren für PU p folgendes Netzwerk:

$$N = \big(V_1 \cup V_2 \cup V_3, (V_1 \times V_2) \cup (V_2 \times V_3) \cup (V_3 \times V_1)\big),$$
$$V_1 = \{p\},\ V_2 = \big\{q \in PU : \exists\, d \in D_p^{\text{fix}}\ \ \mathrm{W}(q, p, \overline{d}\,) > 0\big\},\ V_3 = D_p^{\text{fix}}$$

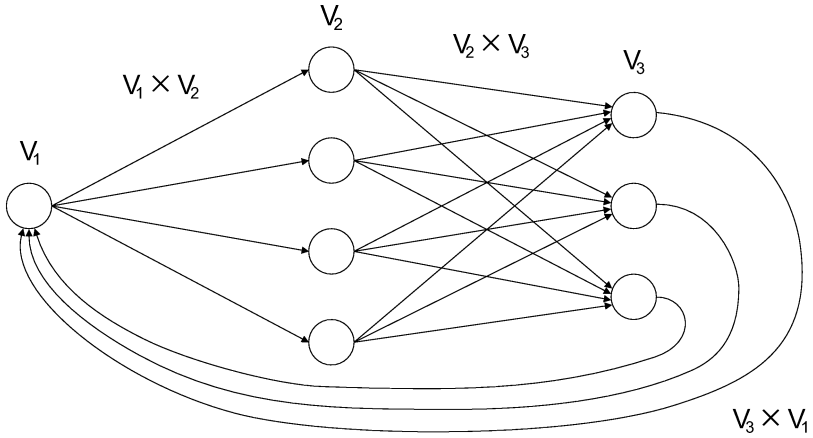


Abbildung 5.1: Netzwerk N für den Fall $|V_2| = 4$, $|V_3| = 3$

Abbildung 5.1 veranschaulicht das Netzwerk N grafisch. Der linke Knoten steht für die zu untersuchende PU p, die mittleren Knoten sind all jene PUs, von denen zu irgendeinem Zeitpunkt des Strategiezeitraumes Trainees zu fixierten Kursen auf PU p kommen, und die rechten Knoten repräsentieren alle von der Optimierung angelegten fixierten Kurse auf PU p. Mit Hilfe dieses Netzwerks wollen wir die Traineezusammensetzungen aller im Strategiezeitraum angelegten fixierten Kurse auf PU p entsprechend den drei Forderungen aus 5.1 umverteilen. Die **Knoten-Kanten-Inzidenzmatrix** $E \in \{+1, -1, 0\}^{V \times A}$ eines Netzwerks (V, A) sei wie folgt definiert:

$$E\big(u, (v, w)\big) = \begin{cases} +1 & \text{falls } u = v \\ -1 & \text{falls } u = w \\ 0 & \text{sonst} \end{cases}$$

Eine **Zirkulation** in einem Netzwerk (V, A) sei ein Vektor $z \in \mathbb{R}^A$, für den $Ez = 0$ erfüllt ist. Durch eine Zirkulation wird jeder Kante eines Netzwerks ein Flusswert zugewiesen. Die Bedingung $Ez = 0$ sichert, dass in jeden Knoten genau so viel hineinfließt wie aus ihm wieder herausfließt. Was genau soll in unserem konstruierten Netzwerk N nun fließen? Es wird die Menge aller Trainees sein, die innerhalb des Strategiezeitraumes auf PU p schulen, nennen wir sie kurz die p**-Trainees**. Am linken Knoten des Netzwerks befinden sich sämtliche p-Trainees. Mittels der $V_1 \times V_2$-Kanten werden sie gesplittet, je nachdem, von welcher Stamm-PU sie kommen. An jedem mittleren Knoten q befinden sich dann jeweils die Trainees, die irgendwann innerhalb des Strategiezeitraumes von PU q nach PU p schulen. Durch die $V_2 \times V_3$-Kanten werden diese Trainees nochmals gesplittet, je nachdem, an welchem konkreten Kurs auf PU p sie teilnehmen. Damit befinden sich in jedem rechten Knoten so viele Trainees wie an dem entsprechenden Kurs teilnehmen. Über die $V_3 \times V_1$-Kanten werden schließlich alle p-Trainees wieder zum linken Knoten zurückgeführt, um die Zirkulation zu vervollständigen. Man mache sich bewusst, dass es sich bei dieser Trainee-Zirkulation nicht um einen realen Vorgang wie beim Pilotenfluss aus dem MIP handelt, sondern dass wir dieses Netzwerk lediglich benutzen, um die Zusammensetzungen der angelegten Kurse nachträglich umzuverteilen.

Die Lösung x von SPTS liefert mit den in 5.1 definierten Größen s und t folgende Zirkulation $\tilde{z}$:

$$\begin{aligned} \forall\,(p,q) \in V_1 \times V_2 \qquad & \tilde{z}_{(p,q)} := s_{q,p} \\ \forall\,(q,d) \in V_2 \times V_3 \qquad & \tilde{z}_{(q,d)} := t_{q,p,d} \\ \forall\,(d,p) \in V_3 \times V_1 \qquad & \tilde{z}_{(d,p)} := x^{\mathrm{C}}_{p,d} \end{aligned}$$

Unser Ziel ist es, die fraktionale Zirkulation $\tilde{z}$ in eine ganzzahlige Zirkulation z umzuwandeln. Die Schranken, bis zu denen jeder Flusswert dabei verändert werden darf, legen wir scharf fest: Ist der Flusswert $\tilde{z}_a$ einer Kante $a \in A$ bereits ganzzahlig, so soll z_a denselben Wert erhalten, ist der Flusswert fraktional, darf höchstens auf- oder abgerundet werden. Beides können wir in der Ungleichung $\lfloor \tilde{z} \rfloor \leq z \leq \lceil \tilde{z} \rceil$ zusammenfassen. Sie ist komponentenweise zu verstehen und beschränkt die Flusswerte aller Kanten. Genauer erfüllt sie über die $V_3 \times V_1$-Kanten die in 5.1 genannte Forderung (1), über die $V_2 \times V_3$-Kanten Forderung (2) und über die $V_1 \times V_2$-Kanten Forderung (3). Damit haben wir alle drei Forderungen aus der Problemstellung auf einen Schlag modelliert. Fassen wir nun alle Zirkulationen, die den drei Forderungen genügen, in der Menge Z zusammen und formen diese äquivalent als Lösungsmenge eines Ungleichungssystems um:

$$Z = \left\{ z \in \mathbb{R}^A : Ez = 0,\ \lfloor \tilde{z} \rfloor \leq z \leq \lceil \tilde{z} \rceil \right\} = \left\{ z \in \mathbb{R}^A : Mz \leq r \right\}$$

$$\text{mit} \quad M = \begin{pmatrix} E \\ -E \\ I \\ -I \end{pmatrix}, \quad r = \begin{pmatrix} 0 \\ 0 \\ \lceil \tilde{z} \rceil \\ -\lfloor \tilde{z} \rfloor \end{pmatrix}$$

Unter den Zirkulationen aus Z wollen wir eine ganzzahlige mit minimalen Rundungsfehlern finden. Es wird sich herausstellen, dass dieses Ziel dank unserer Vorarbeit nun leicht zu erreichen ist.

5.3 Lösung

Eine Matrix heiße **total unimodular**, wenn die Determinante jeder quadratischen Untermatrix den Wert $+1$,-1 oder 0 hat. Unsere Knoten-Kanten-Inzidenzmatrix $E \in \{+1, -1, 0\}^{V \times A}$ enthält in jeder Spalte genau eine $+1$ und eine -1 und ist daher nach dem Satz von Poincaré [5] total unimodular. Damit ist auch Matrix M total unimodular.

Da außerdem die rechte Seite r des Ungleichungssystems $Mz \leq r$ ganzzahlig ist, ist Z nach dem Satz von Hoffmann/Kruskal [5] ein ganzzahliges Polyeder, d.h. alle Ecken haben ganzzahlige Koordinaten. Das Polyeder ist aufgrund der koordinatenweisen Beschränkung selbst beschränkt und wegen $\tilde{z} \in Z$ nichtleer. Weisen wir jeder Kante $a \in A$ des Netzwerks einen Kostenkoeffizienten c_a zu und suchen eine Zirkulation $z \in Z$ mit minimalen Kosten $cz = \sum_{a \in A} c_a z_a$, so wissen wir nach obiger Argumentation sofort, dass stets eine optimale ganzzahlige Lösung existiert.

Unsere Aufgabe ist es nun lediglich, Kostenkoeffizienten so aufzustellen, dass sie genau die auftretenden Rundungsfehler bei der Umwandlung von fixierten zu virtuellen Kursen beschreiben. Gilt für eine Kante a bereits $\tilde{z}_a \in \mathbb{Z}$, kann der Kostenkoeffizient c_a beliebig gewählt werden, da der Flusswert z_a in jeder Zirkulation z wegen $\tilde{z}_a = \lfloor \tilde{z}_a \rfloor \leq z_a \leq \lceil \tilde{z}_a \rceil = \tilde{z}_a$ eindeutig bestimmt ist, also die Kante a immer die gleichen Kosten $c_a z_a$ verursacht. Gilt $\tilde{z}_a \notin \mathbb{Z}$, so wird der Flusswert bei der Umwandlung zur Zirkulation z entweder auf- oder abgerundet. Der Fehler durch Aufrunden beträgt $\lceil \tilde{z}_a \rceil - \tilde{z}_a = \lfloor \tilde{z}_a \rfloor + 1 - \tilde{z}_a$, der durch Abrunden $\tilde{z}_a - \lfloor \tilde{z}_a \rfloor$. Wir setzen den Kostenkoeffizienten c_a so, dass die Kostendifferenz für Auf- und Abrunden genau der Differenz der Rundungsfehler beim Auf- und Abrunden entspricht: $c_a \lceil \tilde{z}_a \rceil - c_a \lfloor \tilde{z}_a \rfloor \stackrel{!}{=} \lfloor \tilde{z}_a \rfloor + 1 - \tilde{z}_a - \tilde{z}_a + \lfloor \tilde{z}_a \rfloor$, also $c_a := 2(\lfloor \tilde{z}_a \rfloor - \tilde{z}_a) + 1$.

Damit haben wir die Aufgabe, alle fixierten Kurse auf PU p in virtuelle umzuwandeln, als kostenminimales Flussproblem $\min\{\, cz : z \in Z \,\}$ formuliert, das sich mit Hilfe polynomieller Verfahren [5] oder des Simplexalgorithmus [6] einfach lösen lässt. Die gesuchten ganzzahligen Kurszusammensetzungen sind dann an den Flusswerten der $V_2 \times V_3$-Kanten der optimalen Zirkulation abzulesen.

Anhang A

Glossar

Aircraft	Flugzeug-Typ
Airline	Fluggesellschaft
Bereederung	Besetzen von Flugzeugen mit Personal
CFG	Condor Fluggesellschaft
Checker	Speziell geschulter CP
COMPAS	Crew Operation Manpower Planning Advanced System
CP	Kapitän (Captain)
CPLEX	Programmpacket von ILOG zum Lösen linearer Optimierungsprobleme und verwandter Probleme [2]
DLH	Deutsche Lufthansa
Einsatzstunde	Stunde, in der ein Pilot fliegerischen oder nichtfliegerischen Dienst leistet
Einsatztag	Tag, an dem ein Pilot Einsatzstunden erbringt
Einstiegsmuster	PUs, auf die NFFs geschult werden
Fixierter Kurs	Kurs, bei dem Startzeitpunkt, Ziel-PU und Teilnehmerzahl gegeben sind
Flotte	Verwaltungsstruktur, bestehend aus Flugzeug-Typ und Fluggesellschaft
FO	Erster Offizier (First Officer)
Fraktionaler Kurs	Fixierter Kurs mit nichtganzzahliger Teilnehmerzahl
Homebase	Flughafen, bei dem ein Pilot stationiert ist
Kurs	Lehrgang, der Piloten fachlich von einer PU auf eine andere qualifiziert
Matching	Einteilen realer Piloten in fixierte Kurse entsprechend ihrer Seniorität
MIP	Gemischt ganzzahliges Problem (Mixed Integer Program)
Muster	Flugzeug-Typ
NFF	Pilot nach der Grundausbildung (Nachwuchsflugzeugführer)
PU	Planungseinheit (Planning Unit); Klasse von Piloten, die durch ein Tripel *Airline-Aircraft-Cockpitfunktion* charakterisiert ist
Ready Entries	Quereinsteiger anderer Fluggesellschaften
Realer Kurs	Virtueller Kurs, bei dem die realen teilnehmenden Piloten gegeben sind

Rückschulung	Type-Rating eines Piloten auf eine bereits durchlaufene PU
Schulung	Alternativbezeichnung für Kurs oder Type-Rating
Schulungsplan	Gesamtheit aller Kurse im Strategiezeitraum
Seniorität	Prioritätssystem für die Kurseinteilung realer Piloten
SFO	Speziell geschulter FO für Langstreckenflüge (Senior First Officer)
SPTS	Langfristige Schulungsplanung für Piloten (Strategic Pilot Training Scheduling)
Stamm-PU	PU, von der ein Trainee wegschult
Stammverteilung	Funktion, die die Herkunftswahrscheinlichkeit eines Trainees angibt
Strategiezeitraum	Zeitraum, für den SPTS durchgeführt werden soll
Subfunktion	Auswahl spezieller Qualifikationen eines Piloten
TraFO	Speziell geschulter FO für Ausbildungszwecke (Training First Officer)
Trainee	Teilnehmer eines Kurses
Type-Rating	Teilnahme eines Trainees an einem realen Kurs
Umlauf	Folge der geleisteten Flüge eines Piloten zwischen zwei Aufenthalten an seiner Homebase
Virtueller Kurs	Fixierter Kurs, bei dem die Stamm-PUs der Trainees gegeben sind
Ziel-PU	PU, auf die ein Trainee schult

Anhang B

Einheitenverzeichnis

[ ]	dimensionslose Einheit
[%]	Prozent (= $\frac{1}{100}$[ ])
[1d]	implizite Einheit Tag (= [d])
[d]	Tag
[€]	Euro
[eh]	Einsatzstunde (= [h])
[et]	Einsatztag (= [d])
[h]	Stunde
[min]	Minute
[p]	Pilot

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] A. Bachem, U. Derigs, M. Jünger, R. Schrader. *Operations Research '93.* Physica-Verlag, 1994

[2] ILOG CPLEX 8.0, *User's Manual.* ILOG, 2002.

[3] Panos Kouvelis, Gang Yu. *Robust Discrete Optimization and Its Applications.* Kluwer Academic Publishers, 1997

[4] R. Tyrrell Rockafellar. *Network Flows and Monotropic Optimization.* John Wiley & Sons, Inc., 1984, S. 101-103

[5] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1999

[6] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions.* Kluwer Academic Publishers, 1998

[7] Ingo Wegener. *Theoretische Informatik – eine algorithmenorientierte Einführung*, 2., durchgesehene Auflage. B. G. Teubner Verlag, 1999

[8] Laurence A. Wolsey. *Integer Programming.* John Wiley & Sons, Inc., 1998

[9] Gang Yu. *Industrial Applications of Combinatorial Optimization.* Kluwer Academic Publishers, Dordrecht, 1998, Chapter 1

[10] Gang Yu. *Operations research in the airline industry.* Kluwer Academic Publishers, Fourth Printing 2002

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit (entsprechend der genannten Verantwortlichkeit) selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Chemnitz, 2004-07-01 Alexander Tschakert