Robuste Terminvergabe im Krankenhaus unter unsicheren Behandlungspfaden

von
Luisa Eickmeyer

Masterarbeit in Mathematik

vorgelegt der

Fakultät für Mathematik, Informatik und Naturwissenschaften der
Rheinisch-Westfälischen Technischen Hochschule Aachen

im September 2013

angefertigt am Lehrstuhl für Operations Research

Erstgutachter:	Prof. Dr. Marco Lübbecke
Zweitgutachter:	Prof. Dr. Arie M.C.A. Koster

Vorwort

Die Terminvergabe in Krankenhäusern ist eine anspruchsvolle Aufgabe, die sich bei einer schlechten Planung auf Maschinen- und Personalkosten sowie auf die Gesundheit von Patienten auswirken kann. Es gibt viele Faktoren, die die Terminplanungen in Krankenhäusern beeinflussen, insbesondere spielen unsichere Ereignisse eine große Rolle. Solche unsicheren Ereignisse können beispielsweise Notfallpatienten oder zusätzliche Behandlungen sein, deren Eintreten zum Planungszeitpunkt noch nicht bekannt waren und die dadurch in einer deterministischen Planung nicht berücksichtigt worden sind. Aus diesen Gründen wird ein robuster Plan zur Terminvergabe benötigt, der in allen erdenklichen Szenarien eine möglichst gute Terminvergabe erstellt. In dieser Arbeit wird dieses Problem behandelt und ein mögliches Lösungsverfahren gezeigt. Dieses Lösungsverfahren beinhaltet ein graphentheoretisches Teilproblem, was in dieser Arbeit zusätzlich für einen allgemeineren Fall betrachtet wird.

Inhaltsverzeichnis

1 Einleitung

Terminvergabe in Krankenhäusern

In Krankenhäusern kommt es häufig zu Wartezeiten, die die Gesundheit von Patienten beeinflussen können. Die Ressourcen des Krankenhauses sind außerdem sinnvoll einzusetzen, damit keine unnötigen Wartezeiten auf teuren Maschinen und keine Überstunden für das Personal entstehen. Dies alles führt für die Krankenhausbetreiber zu hohen Kosten, außerdem ist neben einer gesundheitlichen Beeinträchtigung der Patienten mit einem negativen Ansehen von Krankenhäusern zu rechnen. Aus diesen Gründen sind sinnvolle und möglichst geschickt geplante Terminvergaben unerlässlich. Eine deterministische Terminvergabe reicht häufig nicht aus, um Leerlauf und Verzögerungen zu verhindern. Viele Vorgänge in der Behandlung und Belegung der Patienten in Krankenhäusern ist im Vorhinein oft nicht absehbar oder optimal planbar. Notfallpatienten sind beispielsweise solche Unsicherheitsfaktoren. Ihr Erscheinen ist vorher nicht bekannt und sie müssen oft mit einer höheren Priorität als Patienten, die einen regulären Termin haben, behandelt werden. Dies führt dazu, dass erstellte Pläne nicht mehr gültig sind und Wartezeiten entstehen.

Solche Unsicherheiten, wie etwa Notfallpatienten, kommen im Gesundheitsbereich häufig vor. Weitere Beispiele für solche Unsicherheiten sind etwa stochastische Behandlungszeiten, oder ein bereits erfasster Patient muss aufgrund einer speziellen Diagnose zusätzliche Behandlungen erhalten. Einen Terminplan, der unter allen möglichen Unsicherheiten eine möglichst gute Lösung liefert, nennen wir robust.

In dieser Arbeit betrachten wir eine spezielle Unsicherheit und versuchen unter dieser Unsicherheit einen robusten Plan zur Terminvergabe in Krankenhäusern zu finden. Dazu betrachten wir den Fall, dass alle Patienten für einen Planungszeitraum bekannt sind. Ein Planungszeitraum umfasst dabei beispielsweise einen Tag oder eine Personalschicht. Jeder dieser Patienten muss aufgrund seiner Diagnose in diesem Zeitraum verschiedene Behandlungen in einer bestimmten Reihenfolge durchlaufen. Es kann jedoch sein, dass jeder Patient aufgrund von weiteren Diagnosen während seines Krankenhausaufenthaltes einige Behandlungen zusätzlich zu den im Vorhinein bekannten Behandlungen erhalten muss. Die Behandlungen, die ein Patient insgesamt erhalten wird, werden in einem klinischen Behandlungspfad dargestellt. Wenn nicht alle Behandlungen auf einem Behandlungspfad sicher eintreten werden, nennen wir einen solchen Behandlungspfad in der vorliegenden Arbeit unsicher. Dabei gehen wir davon aus, dass bekannt ist, welche zusätzlichen Behandlungen an welcher Stelle des Behandlungspfads eintreten können.

Mithilfe von mathematischen Optimierungsmethoden lassen sich deterministische Terminpläne für Krankenhäuser erstellen. Um einen robusten Terminplan unter Unsicherheiten zu erstellen, nutzen wir in dieser Arbeit Methoden der robusten Optimierung. Die robuste Optimierung

ist ein Gebiet der Optimierung, in dem bei Unsicherheiten in den Parametern eine für alle Szenarien möglichst stabile Lösung gesucht wird.

Einordnung

Die Basis dieser Arbeit liefert die Planung von Patiententerminen im Krankenhausumfeld. In der Literatur ist dies ein viel behandeltes Thema, da hier ein großes Potenzial in der Einsparung von Kosten und Zeit für Krankenhäuser liegt. Die Planung von Patiententerminen im Krankenhaus unterliegt häufig Unsicherheiten, siehe dazu Schlüchtermann [21] oder Paulussen u. a. [20]. Viele Unsicherheiten wie etwa Patientenausfälle, siehe Liu u. a. [17], wurden bereits behandelt und untersucht. Für die Patientensteuerung unter variablen Behandlungspfaden und stochastischen Behandlungsdauern haben Paulussen u. a. [20] einen Lösungsansatz über ein Multiagentensystem vorgestellt.

Um unter Unsicherheiten zu optimieren, verwenden wir Methoden der robusten Optimierung. Dies ist eine verhältnismäßig neue Methode, um Optimierungsmodelle, deren Parameter mit Unsicherheiten gegeben sind, zu behandeln. Soyster [23] war 1973 einer der ersten, der die robuste Optimierung erforschte, bevor Jahre später Ben-Tal und Nemirovski [4] sowie Ben-Tal und Nemirovski [3] weitere wichtige Erkenntnisse treffen konnten. Allgemein sind in den letzten Jahren viele neue Erkenntnisse für die robuste Optimierung getroffen worden. In dieser Arbeit benutzen wir die Idee der adaptiven robusten Optimierung, den Ben-Tal u. a. [2] im Jahre 2004 entwickelt haben.

Von großer Bedeutung ist in dieser Arbeit das Minimalkostenflussproblem. Edmonds und Karp [10] konnten 1972 zeigen, dass das Minimalkostenflussproblem polynomiell lösbar ist. Wir betrachten eine Abwandlung des Minimalkostenflussproblems in azyklischen Digraphen. In azyklischen Digraphen entspricht dieses Problem einem Maximalkostenflussproblem mit negierten Kantenkosten. Unser Problemabwandlung besteht darin, dass man für das Maximalkostenflussproblem einige Bogenkosten erhöhen darf. In der Literatur ist diese Problematik bislang nach bestem Wissen noch nicht bekannt oder erforscht worden.

Aufbau dieser Arbeit

Diese Arbeit behandelt eine robuste Terminvergabe in Krankenhäusern unter unsicheren Behandlungspfaden. Der Aufbau dieser Arbeit ist dabei wie folgt:
In Kapitel 2 werden wir einige nötige Grundlagen geben, um ein grundlegendes Verständnis und eine übereinstimmende Notation für tiefergehende Betrachtungen und Untersuchungen zu erhalten. Dabei gehen wir knapp auf Komplexitätstheorie, graphentheoretische Notationen und Probleme, das Job Shop Scheduling Problem, ganzzahlige lineare Optimierung, das Colum Generation Verfahren anhand der Dantzig-Wolfe Zerlegung sowie auf robuste Optimierung als Grundlage der Planung in Krankenhäusern ein.

In Kapitel 3 wird das Problem der Terminvergabe unter unsicheren Behandlungspfaden noch einmal genau erläutert und alle Modellannahmen getroffen. Es wird in Abschnitt 3.3 zunächst ein deterministisches Modell zur Terminvergabe gegeben und ausgehend von diesem ein robustes zweistufiges Modell zur betrachteten Unsicherheit erstellt. In Abschnitt 3.4.3 wird die zweite Entscheidungsstufe des robusten Modells als graphentheoretisches Problem auf azyklischen Digraphen interpretiert, dem Maximalkostenflussproblem mit K variablen Kosten, abkürzend K-MKF-Problem genannt.

In Kapitel 4 werden wir ein insgesamtes Lösungsverfahren zur robusten Terminvergabe im Krankenhaus unter unsicheren Behandlungen kennenlernen. Dieses basiert auf dem zweistufigen robusten Modell aus Kapitel 3 . Dabei verzichten wir jedoch zunächst darauf, Lösungsverfahren für das Maximalkostenflussproblem mit K variablen Kosten anzugeben.

Das Maximalkostenflussproblem mit K variablen Kosten untersuchen wir in Kapitel 5 für allgemeine azyklische Digraphen. Dabei unterscheiden wir Digraphen mit nur einer Senke und allgemeine Digraphen mit mindestens zwei Senken. Für Digraphen mit einer Senke finden wir in Abschnitt 5.1 polynomielle Algorithmen, um das Maximalkostenflussproblem mit K variablen Kosten zu lösen. Für das allgemeine Maximalkostenflussproblem mit K variablen Kosten sehen wir in Abschnitt 5.2 ein Optimalitätskriterium.

Lösungsverfahren für das betrachtete Maximalkostenflussproblem mit K variablen Kosten im allgemeinen Fall werden in dem folgenden Kapitel 6 gezeigt. Neben dem Lösen als ganzzahliges lineares Optimierungsproblem werden wir ein Lösungsverfahren über eine genauere Betrachtung der Quelle-Senke-Wege finden sowie ein Lösungsverfahren über das Column Generation Verfahren.

Abschließend werden wir in Kapitel 7 einen knappe Zusammenfassung der Ergebnisse dieser Arbeit liefern. Außerdem werden wir einen Ausblick auf Modifikationen des Maximalkostenflussproblems mit K variablen Kosten und der betrachteten Problemstellung der Terminplanung unter Unsicherheiten in Krankenhäusern geben.

2 Grundlagen und Notationen

Bevor wir uns der genauen Problemstellung der robusten Terminvergabe im Krankenhaus unter unsicheren Behandlungspfaden widmen, wollen wir zunächst in diesem Kapitel einige benötigte Begrifflichkeiten und Hintergründe kurz erläutern und wiederholen sowie eine einheitliche Notation angeben, um die nötige Grundlage für diese Arbeit zu schaffen. Dabei wird im Folgenden auf Komplexität, verwendete Graphenprobleme und ihre Notationen sowie auf Methoden der ganzzahligen linearen Optimierung und dabei insbesondere auf die Dantzig-Wolfe Zerlegung und ihre Anwendung in dem Column Generation Verfahren eingegangen. Außerdem wird das Job Shop Scheduling Problem erläutert, welches für den verwendeten Ansatz der Terminvergabe im Krankenhaus benötigt wird, und die robuste Optimierung umrissen.

2.1 Komplexität

Als Komplexität eines Problems bezeichnet man den Aufwand, der benötigt wird, um dieses Problem zu lösen. Dabei wird der Aufwand über den besten bekannten Algorithmus zur Lösung dieses Problems definiert. Der Aufwand kann in der Anzahl benötigter Rechenschritte als zeitlicher Aufwand oder in dem benötigten Speicherbedarf als Speicheraufwand gemessen werden. Wir beschränken uns in dieser Arbeit auf den zeitlichen Aufwand von Algorithmen. Für die Laufzeit eines Algorithmus wird die maximale Zeit angegeben, die der Algorithmus für die Lösung eines Problems benötigen kann. Die Laufzeit eines Algorithmus A bezeichnen wir mit $t_A(n)$ bei einer Eingabe der Länge $n \in \mathbb{N}$.

Da vor allem der Anstieg des Ressourcenverbrauchs, also die benötigte Zeit bei wachsender Eingabegröße, von Interesse ist, gibt man für die Komplexität eines Algorithmus meist obere oder untere Schranken in Landau-Notation an. Für zwei Funktionen $f, g : \mathbb{R} \to \mathbb{R}$ bedeutet $f \in \mathcal{O}(g)$ oder $f = \mathcal{O}(g)$, dass es ein $x_0 \in \mathbb{R}$ und ein $c > 0$ gibt, sodass $|f(x)| \leq c \cdot |g(x)|$ für alle $x > x_0$ gilt, wobei $|f(x)|$ die Laufzeit der Funktion f beschreibt. Die Funktion f ist also asymptotisch ab einem gewissen Punkt durch ein Vielfaches der Funktion g beschränkt. Für die Laufzeit $t_A(n)$ eines Algorithmus A, die durch eine Funktion f beschrieben wird, schreiben wir auch $t_A(n) = \mathcal{O}(f)$ oder A liegt in $\mathcal{O}(f)$.

Die Laufzeit eines Algorithmus A ist polynomiell beschränkt, falls eine Konstante $\alpha \in \mathbb{N}$ existiert, sodass

$$t_A(n) = \mathcal{O}(n^\alpha)$$

gilt. Man sagt in diesem Fall auch, dass Algorithmus A ein polynomieller Algorithmus ist. Anhand von Schranken kann man Probleme in Komplexitätsklassen einordnen. Bei Problemen, die in derselben Komplexitätsklasse liegen, ist der Aufwand zur Lösung dieser Probleme ähnlich groß. In diese Arbeit werden die beiden Klassen $\mathcal{P}$ und $\mathcal{NP}$ benötigt, es gibt jedoch noch

weitere. In der Klasse $\mathcal{P}$ liegen alle Probleme, für die ein polynomiell beschränkter Algorithmus existiert. In der Klasse $\mathcal{NP}$ liegen alle Probleme, für die Richtigkeit einer gegebenen Lösung in Polynomialzeit geprüft werden kann. Insbesondere bedeutet das, dass alle Probleme aus $\mathcal{P}$ auch in $\mathcal{NP}$ liegen, also gilt $\mathcal{P} \subseteq \mathcal{NP}$. Bislang ist jedoch unklar, ob die beiden Klassen ungleich oder gleich sind.

Ein Problem A heißt auf ein Problem B *polynomiell reduzierbar* , bezeichnet mit $A \leq_p B$, genau dann, wenn es eine polynomielle Abbildung f gibt, sodass die Eingabe von Problem A zu einer Eingabe von Problem B transformiert werden kann.

Nun werden noch die Begriffe der $\mathcal{NP}$-Vollständigkeit und $\mathcal{NP}$-Schwere benötigt. Ein Problem A heißt *$\mathcal{NP}$-schwer* genau dann, wenn $B \leq_p A$ für alle $B \in \mathcal{NP}$ gilt. Das Problem A ist also mindestens genauso schwer zu lösen wie alle Probleme, die in $\mathcal{NP}$ liegen. Falls zusätzlich $A \in \mathcal{NP}$ gilt, ist das Problem A *$\mathcal{NP}$-vollständig*. Zwei Probleme A und B heißen *polynomiell äquivalent*, wenn sowohl $A \leq_p B$ als auch $B \leq_p A$ gilt.

Einen Algorithmus A, dessen Laufzeit polynomiell im Wert der Eingabegröße ist, nennt man auch *pseudopolynomiell*. Der Unterschied zwischen pseudopolynomiellen und polynomiellen Algorithmen ist, dass polynomielle Algorithmen in der Länge der Eingabegröße polynomiell sind. $\mathcal{NP}$-vollständige Probleme, für die ein pseudopolynomieller Algorithmus existiert, nennt man *schwach $\mathcal{NP}$-vollständig*. Probleme, für die kein pseudopolynomieller Algorithmus existiert, nennt man auch *stark $\mathcal{NP}$-vollständig*.

Eigenschaften und Definitionen der Komplexitätstheorie, die über das hier Vorgestellte hinaus gehen, sind zum Beispiel Jongen u. a. [13] oder Garey und Johnson [11] zu entnehmen.

2.2 Graphentheoretische Notationen

In diesem Abschnitt wird nun kurz eine einheitliche Notation von Graphen und ihren benötigten Eigenschaften gegeben, ohne diese vollständig definieren zu wollen. Diese und weiterführende Notationen sind beispielsweise in Diestel [9] zu finden.

Ein *ungerichteter Graph* $G = (V, E)$ besitzt die Knotenmenge $V = V(G)$ sowie die Kantenmenge $E = E(G)$. Die Kardinalitäten dieser Mengen werden in dieser Arbeit durchgehend mit $|V|$ beziehungsweise $|E|$ bezeichnet, um eine Mehrdeutigkeit der Notationen mit dem Job Shop Scheduling zu vermeiden, siehe dazu Abschnitt 2.4. Eine ungerichtete Kante $e \in E(G)$ eines ungerichteten Graphen G, die zwischen zwei Knoten v und w verläuft, wird auch als $e = (v, w)$ oder $e = (w, v)$ bezeichnet.

Ein *gerichteter Graph* oder *Digraph* $D = (V, A)$ besitzt die Knotenmenge $V = V(D)$ sowie die gerichtete Kantenmenge $A = A(D)$. Die Kardinalitäten dieser Mengen werden analog zu denen der ungerichteten Graphen bezeichnet. Eine gerichtete Kante $a \in A(D)$ eines Digraphen D, die von Knoten v zu Knoten w mit v und $w \in V(D)$ verläuft, wird auch als $a = vw$ notiert.

Die Kante $a' = wv$ wäre demnach die gerichtete Kante von Knoten w zu Knoten v, also die *antiparallele Kante* zu a. Gerichtete Kanten werden auch als *Bogen* bezeichnet.

Einfache *Wege* zwischen zwei Knoten v und w in einem beliebigen Graphen $G = (V, E)$ werden als v-w-Weg bezeichnet. Die Länge l eines Weges p ist $l(p) = |p|$ und bei einer Kantengewichtung $w : E(G) \to \mathbb{Z}$ $l(p) = \sum_{e \in p} w(e)$. Anstelle von Gewichten kann eine Kostenfunktion $c : E \to \mathbb{Z}$ auf den Kanten definiert sein. *Gerichtete Kreise* in einem Digraphen D sind, sofern nicht anders bezeichnet, einfach in dem Sinne, dass jeder Bogen $a \in A(D)$ maximal einmal auf dem Kreis C enthalten sein darf. Dies impliziert jedoch, dass Knoten, im Gegensatz zur allgemein bekannten Definition von gerichteten Kreisen, auf einem Kreis mehrfach enthalten sein dürfen.

Wir betrachten in dieser Arbeit in den meisten Fällen *azyklische Graphen*, also kreisfreie Graphen. Die Knoten eines azyklischen Digraphen $D = (V, A)$ können nach Schrijver [22] in linearer Zeit topologisch sortiert werden. Eine *topologische Sortierung* $v_1, \ldots, v_{|V(D)|}$ der Knoten von D ist eine Anordnung, sodass für alle Bogen $v_i v_j \in A(D)$ gilt, dass $i < j$ ist.

Der *Grad* eines Knotens v in einem ungerichteten Graphen $G = (V, A)$ wird mit $d(v)$ bezeichnet, in gerichteten Graphen $D = (V, A)$ wird an einem Knoten $v \in V(D)$ der Ausgangsgrad mit $d^+(v)$ bezeichnet und der Eingangsgrad mit $d^-(v)$. Den *maximale Ausgangsgrad* eines gerichteten Graphen D bezeichnen wir mit $\Delta^+(D)$, den *maximalen Eingangsgrad* mit $\Delta^-(D)$ sowie den *minimale Ausgangs- und Eingangsgrad* mit $\delta^+(D)$ und $\delta^-(D)$. Außerdem wird die positive Nachbarschaft eines Knotens v in einem Graphen D mit $N^+(v)$ oder mit $N^+{}_D(v)$ bezeichnet.

2.3 Graphenprobleme

Ein Großteil dieser Arbeit benötigt die Kenntnis einiger bekannter Graphenprobleme. In Jongen u. a. [13] oder in Diestel [9] lassen sich diese und weitere nachlesen. Hier in diesem Abschnitt möchten wir kurz die benötigten Probleme vorstellen. Diese Probleme umfassen Varianten des kürzesten Wege Problems und verschiedene Arten des Flussproblems.

2.3.1 Kürzeste Wege

Das kürzeste Wege Problem in gerichteten Graphen

Instanz:	Digraph $D = (V, A)$, ein Quellknoten $s \in V(D)$, ein Senkeknoten $t \in V(D)$ sowie eine Kostenfunktion $c : A(D) \to \mathbb{R}^+$
Aufgabe:	Finde einen s-t-Weg mit minimalen Kosten oder zeige, dass kein solcher existiert

Hierbei sind die Kosten eines Weges die Summe seiner Bogenkosten, also $c(P) = \sum_{a \in P} c(a)$ für einen Weg P.

Das kürzeste Wege Problem kann in Graphen ohne negativ gewichtete Kreise mithilfe des Dijkstra-Algorithmus in $\mathcal{O}(|V(D)|^2)$ gelöst werden, der Algorithmus von Moore, Bellman und Ford hingegen spürt zusätzlich eventuell vorhandene negative Kreise in einer Laufzeit von ebenfalls $\mathcal{O}(|V(D)|^2)$ auf. Beide kürzesten Wege Algorithmen sind unter anderem in Jongen u. a. [13] nachzulesen.

Das längste Wege Problem ergibt sich analog zu dem kürzesten Wege Problem, man sucht jedoch nun einen einfachen s-v-Weg mit maximaler Länge. Dabei beschränken wir uns in dieser Arbeit auf azyklische Digraphen.

Das längste Wege Problem in azyklischen Graphen

Instanz:	Azyklischer Digraph $D = (V, A)$, eine Quelle $s \in V(D)$, eine Senke $t \in V(D)$ sowie eine Kostenfunktion $c : A(D) \to \mathbb{R}^+$
Aufgabe:	Finde einen s-t-Weg mit maximalen Kosten oder zeige, dass kein solcher existiert

Allgemein ist dies nicht in polynomieller Zeit berechenbar. Nach Schrijver [22] ist aber bekannt, dass für azyklische Graphen dieses Problem sogar in einer linearer Zeit von $\mathcal{O}(|A(D)|)$ lösbar ist.

2.3.2 Längenbeschränkte kürzeste Wege

Eine Modifikation des kürzesten Wege Problems ist das längenbeschränkte kürzeste Wege Problem, siehe dazu Ziegelmann [25]. Dabei ist auf den Bogen des betrachteten Graphen $D = (V, A)$ nicht nur eine Kostenfunktion $c : A(D) \to \mathbb{N}$ definiert, sondern zusätzlich auch Längen über eine Längenfunktion $l : A(D) \to \mathbb{N}$ definiert. Ein gesuchter kürzester s-t-Weg darf eine gegebene Längenschranke L nicht überschreiten.

Das längenbeschränkte kürzeste Wege Problem

Instanz:	Digraph $D = (V, A)$, ein Quellknoten $s \in V(D)$, ein Senkeknoten $t \in V(D)$, eine Kostenfunktion $c : A(D) \to \mathbb{N}$, eine Längenfunktion $l : A(D) \to \mathbb{N}$ und eine Längenschranke L
Aufgabe:	Finde einen s-t-Weg in D mit minimalen Kosten, der die Längenschranke einhält oder zeige, dass kein solcher existiert

Es ist also ein s-t-Weg p gesucht, für den gilt, dass $l(p) = \sum_{a \in p} l(a) \leq L$ ist und dessen Kosten $c(p) = \sum_{a \in p} c(a)$ minimal sind. Das Problem einen solchen längenbeschränkten kürzesten Weg zu finden ist nach Garey und Johnson [11] schwach $\mathcal{NP}$-vollständig.

Ein pseudopolynomieller Algorithmus hierfür ist eine Erweiterung des Dijkstra-Algorithmus, der sogenannte *Labeling Dijkstra Algorithmus* nach Aneja u. a. [1]. Bei diesem werden anstelle

der kürzesten Distanzen pro Knoten $v \in V(D)$ mehrere Labels $(c_p, l_p)_v$ abgespeichert, wobei p ein s-v-Weg mit Kosten c_p und einer Länge von l_p ist. Der Algorithmus berechnet einen längenbeschränkten kürzesten Weg in einer Zeit von $\mathcal{O}(|V(D)|^2 L)$, wobei L die gegebene Längenschranke ist.

2.3.3 Kostenminimale Flüsse

Gerade das Finden von kostenminimalen Flüsse in Netzwerken ist für diese Arbeit besonders relevant. Hier werden kurz die wichtigsten Notationen und Eigenschaften zu diesen kostenminimalen Flüssen nach Schrijver [22] gegeben.

Sei $D = (V, A)$ ein Digraph mit einer Quelle s und einer Senke t, wobei $s \in V(D)$ und $t \in V(D)$ gilt. Sei zudem $u : A(D) \to \mathbb{Z}$ eine Kapazitätsfunktion. Ein *s-t-Fluss* f unter der Kapazitätsfunktion u ist eine Abbildung $f : A(D) \to \mathbb{R}$, die die folgenden Eigenschaften erfüllt:

(i)	$f(a) \geq 0$	für alle Bogen $a \in A(D)$	
(ii)	$f(\delta^+(v)) = f(\delta^-(v))$	für alle Knoten $v \in V(D) \backslash \{s, t\}$	Flusserhaltsbedingung
(iii)	$f(a) \leq u(a)$	für alle Bogen $a \in A(D)$	Kapazitätsbedingung

Der Wert eines Fluss f wird beschrieben durch

$$value(f) := f(\delta^+(s)) - f(\delta^-(s)) = f(\delta^-(t)) - f(\delta^+(t))$$

Das maximale Fluss Problem

Instanz:	Digraph $D = (V, A)$, eine Quelle $s \in V(D)$, eine Senke $t \in V(D)$ und eine Kapazitätsfunktion $u : A(D) \to \mathbb{Z}^+$
Aufgabe:	Finde einen s-t-Fluss f mit maximalem Wert oder zeige, dass kein solcher existiert

Da u ganzzahlig ist, existiert ein maximaler Fluss mit ganzzahligem Wert. Zudem ist das maximaler Fluss Problem polynomiell lösbar, der wohl bekannteste Algorithmus dazu ist der Ford-Fulkerson-Algorithmus mit einer Laufzeit von $\mathcal{O}(|V(D)||A(D)|^2)$.

Sofern der betrachtete Digraph D zusätzlich eine Kostenfunktion $c : A(D) \to \mathbb{Z}^+$ besitzt, lassen sich für jeden Fluss f auch *Kosten* bestimmen, nämlich

$$c(f) := \sum_{a \in A(D)} c(a) f(a)$$

Das Minimalkostenflussproblem

Instanz:	Digraph $D = (V, A)$, eine Quelle $s \in V(D)$, eine Senke $t \in V(D)$, eine Kapazitätsfunktion $u : A(D) \to \mathbb{Z}^+$ und eine Kostenfunktion $c : A(D) \to \mathbb{Z}^+$
Aufgabe:	Finde einen s-t-Fluss f mit minimalen Kosten oder zeige, dass kein solcher existiert.

Edmonds und Karp [10] zeigten, dass das Problem, einen kostenminimalen Fluss zu finden, polynomiell lösbar ist.

Man kann Flüsse mit minimalen Kosten auch über eine Balancefunktion definieren. Die so definierten Flüsse werden auch b-Flüsse genannt. Sei dazu ein Digraph $D = (V, A)$, eine Kapazitätsfunktion $u : A(D) \to \mathbb{Z}^+$, eine Kostenfunktion $c : A(D) \to \mathbb{Z}^+$ sowie eine Balancefunktion $b : V \to \mathbb{Z}$ mit $\sum_{v \in V} b(v) = 0$ gegeben. Für einen Knoten $v \in V$ bezeichne der Wert $b(v)$ die *Balance* des Knotens v. Falls $b(v) > 0$ gilt, dann ist der Knoten v ein Quellknoten, gilt hingegen $b(v) < 0$, so ist der Knoten v ein Senkeknoten. Ein *b-Fluss* f ist ein gewöhnlicher Fluss f mit der zusätzlichen Eigenschaft, dass

$$b(v) = f(\delta^+(v)) - f(\delta^-(v))$$

für alle Knoten $v \in V$ gilt. Ein minimaler b-Fluss ist ein b-Fluss mit minimalen Kosten.

Das minimale b-Fluss Problem

Instanz:	Digraph $D = (V, A)$, eine Kapazitätsfunktion $u : A(D) \to \mathbb{Z}^+$, eine Kostenfunktion $c : A(D) \to \mathbb{Z}^+$ sowie eine Balancefunktion $b : V(D) \to \mathbb{Z}^+$ mit $\sum_{v \in V} b(v) = 0$
Aufgabe:	Finde einen b-Fluss f mit minimalen Kosten oder zeige, dass kein solcher existiert

Nach Goldberg und Tarjan [12] gibt es einen polynomiellen Algorithmus, der das minimale b-Fluss Problem löst.

2.4 Job Shop Scheduling

Die grundlegende Problemstellung der robusten Terminvergabe im Krankenhaus basiert auf einer Instanz des Job Shop Scheduling Problems. Dieses Planungsproblem gehört zur Klasse des allgemeinen Shop Scheduling Probleme. In diesem Abschnitt werden die für diese Masterarbeit benötigten Grundkenntnisse des Job Shop Scheduling auf einer Grundlage von Brucker [8] vorgestellt.

Ein *allgemeines Shop Scheduling Problem* besteht aus n Jobs $J_1, \ldots, J_n$ und m Maschinen $M_1, \ldots, M_m$. Die Indexmenge $\{1, \ldots, n\}$ wird dabei mit I bezeichnet und die Indexmenge

$\{1, \ldots, m\}$ mit J. Jeder Job J_i für $i \in I = \{1, \ldots, n\}$ besteht aus einer individuellen Anzahl n_i an Operationen $O_{i1}, \ldots, O_{in_i}$, wobei jede Operation O_{ij} eine Prozessdauer von p_{ij} besitzt und zu jede Operation eine Menge $\mu_{ij} \in \{M_1, \ldots, M_m\}$ an Maschinen gehört, auf denen diese Operation ablaufen kann. Es können optional für jeden Job J_i eine Releasezeit r_i sowie eine Deadline d_i für $i \in I$ definiert sein. Die Releasezeit entspricht dem frühesten Startzeitpunkt, ab wann der Job verfügbar ist, wohingegen die Deadline d_i dem letzten Zeitpunkt entspricht, bis zu dem Job J_i vollständig behandelt werden muss. Auf einer Maschine kann zu jedem Zeitpunkt nur maximal ein Job bearbeitet werden und jeder Job kann offensichtlich zu jedem Zeitpunkt nur auf einer Maschine bearbeitet werden. Ziel ist es, unter einer einer bestimmten Zielfunktion einen zulässigen Plan zu erstellen.

In dem hier zugrunde liegenden *Job Shop Scheduling Problem*, einem Spezialfall des allgemeinen Shop Scheduling Problems, existieren zusätzlich zwischen den jeweiligen Operationen eines Jobs Vorgängerbeziehungen, auch Präzedenzrelationen genannt, die für einen Job J_i in der Form

$$O_{i1} \to O_{i2} \to O_{i3} \to \ldots \to O_{in_i}$$

auftreten, wobei n_i die Anzahl an Operationen von Job J_i ist. Diese Beziehungen müssen eingehalten werden, was bedeutet, dass eine Operation nicht beginnen kann, bevor ihre Vorgängeroperation beendet wurde. Mit C_i bezeichnet man für einen Job J_i, $i \in I$, den Fertigstellzeitpunkt, zu dem die letzte Operation dieses Jobs beendet wird.

Es wird angenommen, dass die Prozesszeiten ganzzahlig sind. Ein Plan ist *zulässig*, falls sich auf keiner Maschine Zeitintervalle überschneiden und wenn sich für einen Job ebenfalls keine Zeitintervalle der einzelnen Operationen überschneiden. Ein Plan ist *optimal*, falls er zulässig ist und eine bestimmte Zielfunktion minimiert beziehungsweise maximiert.

Als *Zielfunktion* kommen verschiedene Möglichkeiten in Frage, etwa das die gesamte Produktionsdauer, der sogenannte Makespan, minimiert werden soll. In dieser Arbeit werden wir uns jedoch darauf beziehen, dass die Endzeitpunkte aller Jobs in der Summe minimal sind, also

$$\min \sum_{i \in I} C_i,$$

wobei $I = \{1, \ldots, n\}$ die Indexmenge aller Jobs ist. Da die Prozessdauern für die Operationen als Parameter festgelegt sind, ist die Zielfunktion äquivalent dazu, die Starttermine der letzten Operationen aller Jobs zu minimieren, also

$$\min \sum_{i \in I} t_{O_{in_i}},$$

wenn $t_{O_{ij}}$ den Startzeitpunkt von Operation O_{ij} beschreibt. Wir vernachlässigen hier die Releasezeiten und Deadlines für alle Jobs. Somit ergibt sich folgende Problemstellung:

Das Job Shop Scheduling Problem

Instanz:	n Jobs $J_1, \ldots, J_n$, wobei jeder Job J_i aus n_i Operationen besteht, Reihenfolgen $O_{i1} \to O_{i2} \to O_{i3} \to \ldots \to O_{in_i}$ für jeden Job $i \in I$, m Maschinen $M_1, \ldots, M_n$, eine Prozesszeit $p_{ij} \in \mathbb{Z}^+$ für jede Operation O_{ij} und eine Maschinenmenge $\mu_{ij} \subset \{M_1, \ldots, M_m\}$ für jede Operation O_{ij} mit $i \in I$ und $j \in J$
Aufgabe:	Finde einen zulässigen Plan x, der $\sum_{i \in I} C_i$ minimiert oder zeige, dass kein solcher existiert

Zu bemerken ist, dass das Job Shop Scheduling Problem allgemein nach Brucker [8] $\mathcal{NP}$-vollständig ist. Es gibt spezielle Instanzen, für die man in polynomieller Zeit einen optimalen Plan finden kann, diese sind jedoch hier nicht weiter von Interesse, da sie in der vorliegenden Arbeit keine Anwendung finden.

2.4.1 Disjunktive Graphen

Mithilfe sogenannter *disjunktiver Graphen* lassen sich Pläne für das Job Shop Scheduling, jedoch auch für andere Shop Scheduling Probleme, darstellen. In der von uns betrachteten Instanz enthält die Menge aller für das Problem zulässigen Pläne einen optimalen Plan, welchen man mithilfe des disjunktiven Graphen finden kann.

(2.1) Definition (Disjunktiver Graph)
Für die von uns gegebene Instanz des Job Shop Scheduling Problems sei der disjunktive Graph $G = (V, C, F)$ wie folgt definiert:

Für jede Operation der Jobs $J_1, \ldots, J_n$ existiert ein Knoten und zusätzlich ein Quellknoten s, also

$$V := \{v_{O_{ij}} : O_{ij} \text{ ist eine Operation von Job } J_i \text{ für alle } i \in I\} \cup \{s\}.$$

Jeder Knoten $v \in V \backslash \{s\}$ besitzt ein Gewicht, was der Prozesszeit p_{ij} der entsprechenden Operation O_{ij} für ein $i \in I$ und $j \in J$ entspricht. Der Quellknoten s erhält ein Gewicht von 0. Die Bogenmenge des Graphen wird durch zwei Mengen C und F beschrieben.

Die Bogenmenge C enthält alle *konjunktiven Bogen*. Diese gerichteten Bogen beschreiben die Vorgängerbeziehungen der Operationen. Für einen Job J_i, $i \in I$, mit den Operationen O_{ij_1} und O_{ij_2} und der Reihenfolge $O_{ij_1} \to O_{ij_2}$ existiert also ein Bogen $v_{O_{ij_1}} v_{O_{ij_2}}$. Zusätzlich werden Bogen von der Quelle s zu jeder ersten Operation O_{i1} für alle Jobs J_i mit $i \in I$ hinzugefügt.

Die ungerichteten *disjunktiven Bogen* sind in der Menge F enthalten. Diese Bogen beschreiben die Maschinenzugehörigkeit. Für je zwei Operationen, die auf derselben Maschine zu absolvieren sind, existiert ein solcher ungerichteter Bogen. ◇

Angenommen, man habe 3 Jobs J_1, J_2 und J_3 sowie 4 Maschinen $M_1, \ldots, M_4$ zur Verfügung und die Jobs besitzen folgenden Operationen:

Job	Operation 1	Operation 2	Operation 3	Operation 4
Job J_1:	M_1, $p_{11} = 2$	M_3, $p_{12} = 1$	M_2, $p_{13} = 1$	M_4, $p_{14} = 2$
Job J_2:	M_2, $p_{21} = 3$	M_1, $p_{22} = 2$	M_4, $p_{23} = 2$	M_3, $p_{24} = 1$
Job J_3:	M_2, $p_{31} = 2$	M_3, $p_{32} = 2$	M_1, $p_{33} = 3$	M_4, $p_{34} = 2$

Tabelle 2.1: Beispiel Job Shop Scheduling

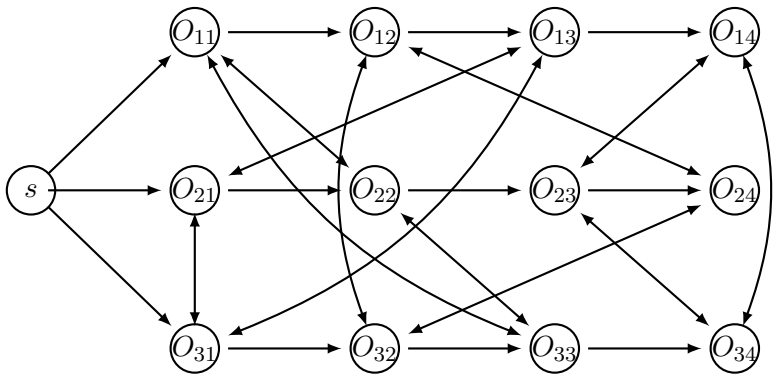


Abbildung 2.1: Beispiel eines disjunktiven Graphen zu Tabelle 2.1

Abbildung 2.1 zeigt den disjunktiven Graphen, der durch die Daten aus Tabelle 2.1 definiert ist. Die Knotenkosten sind durch die Prozessdauern der entsprechenden Operationen aus Tabelle 2.1 gegeben.

In einem zulässigen Ablaufplan für das Job Shop Scheduling Problem existieren nicht nur Reihenfolgen zwischen den Operationen der Jobs, sondern auch Reihenfolgen zwischen den Jobs, die auf einer Maschine laufen. Für den disjunktiven Graphen bedeutet das, dass die disjunktiven Bogen gerichtet werden müssen, um einen zulässigen Plan zu erhalten. Dies lässt sich dadurch gewährleisten, dass der Graph, der durch das Richten von Bogen entsteht, azyklisch sein muss. Damit ist für jede Maschine eine eindeutige Reihenfolge festgelegt.

Disjunktive Bogen, die gerichtet wurden nennt man *fixiert* und eine *Auswahl* $\mathcal{S}$ ist eine Menge fixierter disjunktiver Bogen. Eine Auswahl $\mathcal{S}$ ist *vollständig*, wenn jeder disjunktive Bogen fixiert wurde und der so entstehende Graph $G(S) = (V, C \cup S)$ azyklisch ist.

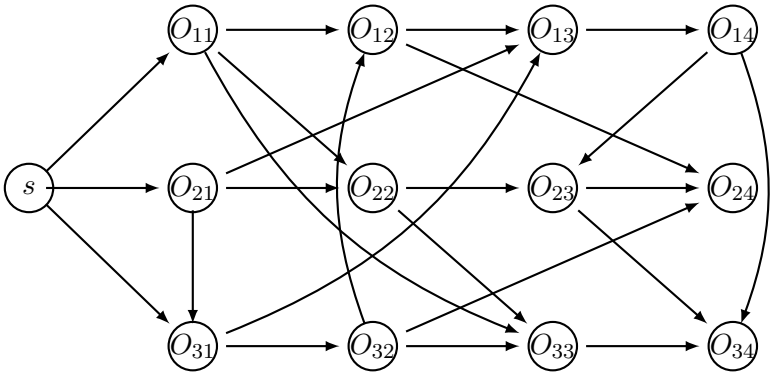


Abbildung 2.2: Beispiel einer vollständigen Auswahl eines disjunktiven Graphen

Abbildung 2.2 ist ein Beispiel für eine vollständige Auswahl anhand des zuvor gewählten Beispiels eines disjunktiven Graphen aus Abbildung 2.1.

Eine vollständige Auswahl beschreibt einen zulässigen Plan x, siehe dazu Brucker [8]. Dies bedeutet insbesondere, dass immer eine vollständige Auswahl eines disjunktiven Graphen existieren wird, die einen optimalen Plan darstellt.

Unsere gewählte Zielfunktion $\min \sum_{i \in I} C_i$, bedeutet wie schon beschrieben, dass die addierten Endzeitpunkte der letzten Operationen für alle Jobs minimiert werden müssen. Um einen Endzeitpunkt C_i zu berechnen, muss ein längster s-$v_{O_{in_i}}$-Weg von der Quelle s zu dem Knoten der letzten Operation O_{in_i} eines Jobs J_i berechnet werden. Dies kann als kostenminimales Flussproblem aufgefasst werden.

Das Job Shop Scheduling Problem über disjunktive Graphen

Instanz:	Ein disjunktiver Graph $G = (V, C, F)$ einer Job Shop Scheduling Probleminstanz
Aufgabe:	Finde eine vollständige Auswahl $\mathcal{S}$, die $\min \sum_{i \in I} C_i$ minimiert oder zeige, dass keine solche existiert

2.5 Ganzzahlige lineare Optimierung

Da in dieser Masterarbeit ganzzahlige lineare Optimierungsprobleme modelliert werden, werden hier in diesem Abschnitt kurz die wichtigsten Eigenschaften, die hier benötigt werden, zusammengefasst. Näheres lässt sich Schrijver [22] entnehmen.

Ein *Polyeder* $\mathcal{P}$ ist eine Teilmenge des $\mathbb{R}^n$ und wird durch eine endliche Anzahl an linearen Ungleichungen beschrieben, also $\mathcal{P} = \{x \in \mathbb{R}^n | Ax \leq b\}$ mit $A \in \mathbb{R}^{mxn}$ und $b \in \mathbb{R}^m$. Das Polyeder $\mathcal{P}$ ist konvex. Ein Punkt $x \in \mathcal{P}$ ist ein Extrempunkt von $\mathcal{P}$, also ein Eckpunkt des

Polyeders, falls er nicht als Konvexkombination von Punkten in $\mathcal{P}$ dargestellt werden kann. Das bedeutet, dass jeder innere Punkt y von $\mathcal{P}$ in folgender Weise dargestellt werden kann:

$$y = \sum_{i=1}^{K} \lambda^i y^i \text{ mit } \lambda^i \in \mathbb{R}^+ \text{ und } \sum_{i=1}^{K} \lambda^i = 1,\ y^i \in \mathcal{P}\backslash\{y\} \text{ für alle } i = 1, \ldots, K$$

Lineare Optimierungsprobleme sind Probleme, die eine lineare Zielfunktion in einem Polyeder minimieren oder maximieren. Sie haben also für $c \in \mathbb{R}^n$ eine der beiden folgenden Formen

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax \leq b \end{aligned} \qquad\qquad \begin{aligned} \max \quad & c^T x \\ s.t. \quad & Ax \leq b. \end{aligned}$$

Mit dem *Dualitätssatz* für lineare Programme lassen sich Probleme umformulieren. Sei dazu $A \in \mathbb{R}^{mxn}, b \in \mathbb{R}^m$ und $c \in \mathbb{R}^n$. Dann gilt

$$\max\{c^T x | Ax \leq b\} = \min\{b^T y | A^T y = c, y \geq 0\},$$

sofern mindestens eins der beiden linearen Programme endlich ist.

Für ein lineares Optimierungsproblem ist jeder Punkt $x \in \mathcal{P}$ eine zulässige Lösung. Ist $\mathcal{P}$ leer, so ist das Optimierungsproblem unzulässig. Eine zulässige Lösung $x \in \mathcal{P}$, die unter einer Zielfunktion $c \in \mathbb{R}^n$ den besten Wert annimmt, nennt man eine optimale Lösung. Lineare Programme können mithilfe der Ellipsoidmethode in polynomieller Zeit gelöst werden, in der Praxis wird das Problem jedoch oft über das Simplexverfahren gelöst.

Ein lineares Programm ist ein *ganzzahliges lineares Programm*, falls die Variable x ganzzahlig gefordert ist, also beispielsweise

$$\max\{c^T x | Ax \leq b, x \in \mathbb{Z}^n\}.$$

Es gilt

$$\max\{c^T x | Ax \leq b, x \in \mathbb{Z}^n\} \leq \max\{c^T x | Ax \leq b\}$$

und bislang sind für das Lösen ganzzahliger linearer Optimierungsmodelle keine polynomiellen Algorithmen bekannt. Man kann sogar zeigen, dass das Lösen von ganzzahligen linearen Programmen $\mathcal{NP}$-vollständig ist. Desweiteren sind nicht nur rein ganzzahlige Probleme, sondern auch $x \in \mathbb{Z}^K \times \mathbb{R}^L$ mit $K + L = n$ möglich. Diese sind auch als gemischt-ganzzahlige lineare Programme bekannt.

Ein rationales Polyeder $\mathcal{P}$ ist *ganzzahlig*, genau dann wenn für alle $c \in \mathbb{Q}^n$ das lineare Programm $\max\{c^T x : x \in \mathcal{P}\}$ einen ganzzahligen Wert annimmt.

Eine wichtige Eigenschaft um mit ganzzahligen Polyedern zu arbeiten, ist die totale Unimodularität der Nebenbedingungsmatrix $A \in \mathbb{R}^{m \times n}$. Eine Matrix $A \in \mathbb{R}^{m \times n}$ ist genau dann total unimodular, wenn jede quadratische Untermatrix eine Determinante von 0, 1 oder -1 besitzt. Insbesondere gilt dann, dass die *lineare Relaxierung* $\max\{c^T x | Ax \leq b\}$ eines ganzzahligen linearen Programms $\max\{c^T x | Ax \leq b, x \in \mathbb{Z}^n\}$ immer eine ganzzahlige Lösung annimmt. Beispiele für total unimodulare Matrizen sind etwa die Inzidenzmatrix eines Digraphen, Einheitsmatrizen oder die Nebenbedingungsmatrix eines minimale Kosten Fluss Problems.

Es gibt natürlich auch Optimierungsprobleme, die über einem Polyeder eine quadratische Zielfunktion minimieren oder maximieren wollen. In dieser Arbeit soll jedoch hauptsächlich mit Methoden der linearen Optimierung gearbeitet werden, weswegen wir hier von einer näheren Betrachtung von diesen sogenannten *quadratischen Optimierungsproblemen* absehen.

Für die Bezeichnung von linearen oder quadratischen Programmen verwenden wir als einheitlichen Begriff Modell. Da wir einige nichtlineare Formulierungen angeben werden, soll dies ein grober Oberbegriff für diese Programme sein.

2.6 Dantzig-Wolfe Zerlegung und Column Generation

Das Lösen ganzzahliger linearen Programme ist im Allgemeinen $\mathcal{NP}$-vollständig, falls nicht bestimmte Bedingungen wie etwa totale Unimodularität der Ungleichungsmatrix A gelten und somit das lineare Programm gelöst werden darf. Im Allgemeinen werden solche ganzzahligen Probleme mittels eines Branch-und-Bound-Algorithmus gelöst. Hier in diesem Abschnitt werden wir anhand von Lübbecke und Desrosiers [18] ein weiteres Lösungsverfahren von ganzzahligen Optimierungsproblemen auf der Grundlage der *Dantzig-Wolfe Zerlegung* zeigen.

2.6.1 Column Generation

Das Verfahren der Column Generation, also der Spaltengenerierung, beruht darauf, dass man zur Lösung eines linearen Programms

$$\begin{aligned} z^* := \min \quad & \sum_{j \in J} c_j x_j \\ s.t. \quad & \sum_{j \in J} a_j x_j \leq b \\ & x_j \geq 0 \qquad\qquad\qquad\qquad j \in J, \end{aligned}$$

dem sogenannten *Masterproblem* [MP], nicht alle möglichen Variablen x_j mit $j \in J$ betrachtet, da $|J|$ sehr groß sein kann. Stattdessen beschränkt man die Betrachtung auf eine Teilmenge $J' \subseteq J$. Dies führt zu einem eingeschränkten Masterproblem, auch *beschränktes Masterproblem* [RMP] genannt. Bei der Lösung des RMPs durch den Simplexalgorithmus wird in jedem einzelnen Schritt nach Nichtbasisvariablen x_j gesucht, deren reduzierte Kosten $\overline{c}_j$ den aktuellen Zielfunktionswert c verbessern. Im Pricingschritt wird also mithilfe der Dualvariablen u

$$\arg\min\{\overline{c}_j := c_j - u^T a_j | j \in J\}.$$

gesucht. Seien nun $\overline{x}$ und $\overline{u}$ die optimalen primalen und dualen Lösungen des RMPs. Dann ist das Pricingproblem in diesem Falle

$$\overline{c}^* := \min\{c(a) - \overline{u}^T a | a \in \mathcal{A}\},$$

wobei $\mathcal{A}$ die Menge der Spalten a_j mit $j \in J$ ist und die Kosten $c(a)$ anhand der Spalte a_j berechnet werden können. Ist $\overline{c}^* \geq 0$, so sind alle reduzierten Kosten $\overline{c}_j$ negativ für $j \in J$ und damit löst $\overline{x}$ das MP optimal. Ansonsten fügen wir der Matrix des RMPs eine Spalte a_j^* hinzu und den entsprechenden Kostenkoeffizienten der Zielfunktion c. Anschließend reoptimieren wir das RMP. Dies ist das Column Generation Verfahren.

Es ergibt sich für den optimalen Zielfunktionswert z^* des MPs dabei die Schranken

$$\overline{z} + \overline{c}^* \leq z^* \leq \overline{z},$$

wobei $\overline{z}$ der optimale Zielfunktionswert des RMPs ist. Bei Maximierungsproblemen muss man analog prüfen, ob $\overline{c}^* > 0$ ist. Das Column Generation Verfahren ist algorithmisch aufgeschrieben das Folgende:

Algorithmus 1 Column Generation

Initialisiere das RMP mit einer sinnvollen Teilmenge;
repeat
  Löse das RMP optimal und erhalte die primal optimale Lösung $\overline{x}^*$ und die dual optimale Lösung $\overline{u}^*$;
  Löse das Pricingproblem und erhalte dafür die optimale Lösung $\overline{c}^*$;
  if $\overline{c}^* < 0$ **then**
    Sei $j \in J$ mit $\overline{c}^* = \overline{c}_j$;
    Füge die Spalte (c_j, A_j) und die Variable x_j in das RMP ein;
  end
until $\overline{c}^* \geq 0$;

2.6.2 Dantzig-Wolfe Zerlegung

Die Dantzig-Wolfe Zerlegung ist eine Möglichkeit, ein lineares Programm mit einer speziellen Struktur zu lösen. Das Optimierungsmodell liegt dabei in der Form

$$\begin{aligned} z^* := \min \quad & c^T x \\ s.t. \quad & Ax \leq b \\ & Dx \leq d \\ & x \geq 0 \end{aligned}$$

vor, wobei die Nebenbedingungen in folgender Struktur auftreten:

$$\begin{pmatrix} A^1 & A^2 & A^3 & \dots & A^k \\ D^1 & & & & \\ & D^2 & & & \\ & & D^3 & & \\ & & & \ddots & \\ & & & & D^\kappa \end{pmatrix} x \leq \begin{pmatrix} b \\ d^1 \\ d^2 \\ d^3 \\ \vdots \\ d^\kappa \end{pmatrix},$$

D^k und d^k liegen in passender Größe zueinander vor. Die Nebenbedingung

$$A^1x^1 + A^2x^2 + A^3x^3 + \ldots + A^\kappa x^\kappa \leq b$$

ist die gemeinsame Nebenbedingung und die übrigen Nebenbedingungen

$$D^k x \leq d^k \text{ für alle } k = 1, \ldots, \kappa$$

sind einzelne Nebenbedingungen für $k \leq \kappa$. Zu den einzelnen Nebenbedingungen lassen sich die Polyeder $P^k := \{x | D^k x \leq d^k, x \geq 0\}$ definieren. Jeder Punkt $x \in P^k$ kann, sofern $P^k \neq \emptyset$ und beschränkt ist, als Konvexkombination von Extrempunkten von P^k für $k = 1, \ldots, \kappa$ beschrieben werden, und im Falle eines unbeschränkten Polyeders als Konvexkombination von Extrempunkten $\{p_q^k\}_{q \in Q^k}$ und einer gewichteten Kombination von Extremvektoren $\{p_r^k\}_{r \in R^k}$ für $k \leq \kappa$ dargestellt werden. Q^k ist die Indexmenge zu den Extrempunkte und die Menge R^k die Indexmenge zu allen Extremvektoren mit $k \leq \kappa$. Eingesetzt in das eigentliche Modell mit $c_j^k = c^T p_j^k$ und $a_j^k = A p_j^k$ ergibt dies das *Masterproblem*

$$
\begin{aligned}
z^* := \min \quad & \sum_{k=1}^{\kappa} \Big(\sum_{q \in Q^k} c_q^k \lambda_q^k + \sum_{r \in R^k} c_r^k \lambda_r^k \Big) \\
s.t. \quad & \sum_{k=1}^{\kappa} \Big(\sum_{q \in Q^k} a_q^k \lambda_q^k + \sum_{r \in R^k} a_r^k \lambda_r^k \Big) \leq b && [u] \\
& \sum_{q \in Q^k} \lambda_q^k = 1 \quad \forall k = 1, \ldots, \kappa && [v] \\
& \lambda_q^k \geq 0 \quad \forall q \in Q^k, k = 1, \ldots, \kappa,
\end{aligned}
$$

Die Anzahl an Variablen ist hierbei sehr groß, da alle Extrempunkte und Extremvektoren betrachtet werden müssen. Falls $x \equiv 0$ eine zulässige Lösung in $\mathcal{P}$ ist, kann die Konvexitätsbedingung $\sum_{q \in Q^k} \lambda_q = 1$ durch $\sum_{q \in Q^k} \lambda_q \leq 1$ mit $k \leq \kappa$ ersetzt werden. Die Dualvariablen werden durch u und v beschrieben.

Eine optimale Lösung besteht nur aus einem einzelnen Extrempunkt. Da wir jedoch nicht jeden möglichen Extrempunkt betrachten wollen, wenden wir zur Lösung das Column Generation Verfahren an.

Es entstehen nun κ Pricingprobleme der Dantzig-Wolfe Zerlegung. Seien dazu $\overline{u}$ und $\overline{v}$ die optimalen Dualvariablen des dazugehörigen RMPs. Die Pricingprobleme haben nun folgende Struktur:

$$
\overline{c}^{k*} := \min\{(c^{kT} - \overline{u}^T A^k) x^k - \overline{v}^k | D^k x^k \leq d^k, x^k \geq 0\} \quad k \in \{1, \ldots, \kappa\}
$$

Durch die Anwendung des Column Generation Verfahrens erhält man eine Schranke von

$$
\overline{z}^* + \sum_{k \leq \kappa} \overline{c}^{k*} \leq z^* \leq \overline{z},
$$

wobei c^{k*} der optimale Wert des Pricingproblems für $k = 1, \ldots, \kappa$ ist, $\overline{z}$ der optimale Zielfunktionswert des RMPs und z^* der optimale Wert des Masterproblems. Bei der Anwendung des Column Generation Verfahrens verfolgt man das Vorgehen von Algorithmus **??**.

2.7 Robuste Optimierung

Sei ein lineares Programm

$$\min\{c^T x | Ax \leq b\}$$

gegeben, wobei $x \in \mathbb{R}^n$ dem Vektor der Entscheidungsvariablen und $c \in \mathbb{R}^n$ dem Zielfunktionskoeffizientenvektor entspricht. Die Matrix $A \in \mathbb{R}^{m \times n}$ beschreibt die Nebenbedingungsmatrix und der Vektor $b \in \mathbb{R}^m$ entspricht der rechten Seite der Nebenbedingungen. Unter bestimmten Umständen treten diese Werte nicht als feste Werte auf, sondern können Unsicherheiten unterliegen. Da die Werte A, b und c Zufallsvariablen entsprechen, ist die stochastische Optimierung ein klassischer Ansatz solch unsichere Probleme zu lösen. Dies wollen wir aber in dieser Arbeit vernachlässigen. Stattdessen wollen wir Ansätze von Ben-Tal und Nemirovski [4] sowie Ben-Tal u. a. [2] auf das Problem der Terminvergabe im Krankenhaus übertragen und damit Lösungsverfahren bestimmen.

Zu jeder möglichen Unsicherheit gehören individuelle Werte der Eingabedaten A, b und c. Sei $\mathcal{S}$ die Menge aller möglichen Unsicherheiten, dann gibt es für jedes $S = (c, A, b) \in \mathcal{S}$ die entsprechenden Unsicherheitsparameter A^S, b^S sowie c^S. Eine einzutretende Unsicherheit wird auch ein *Szenario* genannt. Wir betrachten in diesen Fall das Problem, dass in allen möglichen Szenarien die Nebenbedingung $Ax \leq b$ eingehalten werden soll.

Die robuste Optimierung kann in mehreren Stufen betrieben werden. Zunächst betrachten wir die robuste Optimierung als einstufiges Modell, das heißt, dass alle Entscheidungen vor dem Bekanntwerden des Szenarios getroffen werden.

In einem *deterministischen Optimierungsproblem* der Form $\min\{c^T x | Ax \leq b\}$ sind alle benötigten Daten (c, A, b) mit Sicherheit gegeben und bekannt. Für diese Arbeit betrachten wir jedoch keine deterministischen Modelle, sondern Probleme, die unter Unsicherheit vorliegen. Ein *unsicheres lineares Optimierungsproblem*

$$\{\min\{c^T x | Ax \leq b\}\}_{(c,A,b)\in\mathcal{S}}$$

ist eine Menge an linearen Optimierungsproblemen, wobei die Eingabedaten durch eine Unsicherheitsmenge $\mathcal{S}$ gegeben sind. Die Menge $\mathcal{S}$ wird auch die *Szenarienmenge* genannt.

Für ein unsicheres lineares Optimierungsproblem ist $x \in \mathbb{R}^n$ eine *robust zulässige Lösung*, falls

$$Ax \leq b \text{ für alle } (c, A, b) \in \mathcal{S}$$

gilt. Für eine robust zulässige Lösung x müssen also alle Nebenbedingungen in allen möglichen Szenarien gelten. Den Wert, den ein unsicheres Optimierungsproblem im schlimmsten Fall annehmen kann, ist der *robuste Wert* $\hat{c}(x)$. Dies entspricht dem größten Wert, den die unsichere Zielfunktion c über alle möglichen Szenarien annehmen kann, also

$$\hat{c}^T(x) := \max_{(c,A,b)\in\mathcal{S}} \{c^T x\}.$$

Dieser Wert wird auch der *Worst-Case-Wert* oder der *Max-Case-Wert* genannt.

Der *robuste Counterpart* eines unsicheren linearen Optimierungsproblems ist

$$\min\{y | (c^S)^T(x) \leq y, Ax \leq b \text{ für alle } (c, A, b) \in \mathcal{S}\}.$$

Eine optimale Lösung x des robusten Counterparts ist eine *robust optimale Lösung* und der optimale Wert $y = \hat{c}(x)$ ist ein *robust optimaler Wert*. Eine robuste optimale Lösung x gibt also den kleinsten Wert an, den eine Lösung des unsicheren Optimierungsproblems im schlechtesten Szenario annimmt.

Das Modell des Robust Counterparts lässt sich umformulieren zu

Modell 1

$$\begin{aligned} \min_{x\in\mathcal{X}} \max_{S\in\mathcal{S}} \quad & (c^S)^T x & \\ s.t. \quad & A^S x \leq b^S & \forall S \in \mathcal{S} \\ & x \in \mathcal{X}, & \end{aligned}$$

wobei $\mathcal{X}$ die Menge aller robust zulässigen Lösungen definiert.

Als *Worst-Case-Szenario* oder auch *Max-Case-Szenario* bezeichnen wir für eine gegebene robust zulässige Lösung x das Problem

$$\begin{aligned} \max_{S\in\mathcal{S}} \quad & (c^S)^T x & \\ s.t. \quad & A^S x \leq b^S & \forall S \in \mathcal{S} \end{aligned}$$

Hierbei ist x fest gegeben und man sucht für diesen Wert das Szenario, welches den schlechtesten Zielfunktionswert liefert. Dieses Modell nennen wir auch das *Max-Szenario-Problem* oder den *Max-Szenario-Fall*.

Einstufige Entscheidungen bedeuten, dass einmal getroffene Entscheidungen nicht revidiert werden können. Es gibt weitere, mehrstufige Ansätze, die in dieser Hinsicht weniger konservativ sind. Beispiele sind etwa Ansätze der Wiederherstellung, „Recoverability “, siehe dazu Liebchen u. a. [16] oder der adaptiven robusten Optimierung nach Ben-Tal u. a. [2]. Auch in dieser Arbeit verfolgen wir einen mehrstufigen Ansatz mit der Idee der adaptiven robusten Optimierung, die jedoch auch dazu genutzt werden kann, Lösungen nach Eintreten des tatsächlichen Szenarios zu verbessern.

Bei der einstufigen robusten Optimierung werden alle Entscheidungen vor Eintreten eines Szenarios festgelegt und anschließend nicht mehr verändert. Bei der Wiederherstellung von Lösungen können Entscheidungen nach Eintreten eines Szenarios noch geringfügig verändert werden, um den robust optimalen Wert zu verbessern. Bei der adaptiven robusten Optimierung werden nicht alle Entscheidungen vor dem tatsächlichen Eintreten eines Szenarios getroffen, sondern manche Variablen werden erst nach Gewissheit über das Szenario festgelegt. Dabei werden die Entscheidungsvariablen in zwei Mengen unterteilt: diejenigen, die im Vorhinein festgelegt werden, als *unveränderliche Variablen* und die *veränderlichen Variablen*, die nach Eintreffen des Szenarios neu bestimmt werden können. Natürlich können Variablen auch zu anderen Zeitpunkten bestimmt werden, wir befassen uns in dieser Arbeit jedoch nur mit einem maximal zweistufigen Fall.

Dazu schreiben wir das unsichere lineare Optimierungsmodell mit $x = (u, v)$ um zu

$$\{\min_{u,v,y}\{y | c^T(u,v) \leq y, A(u,v) \leq b\}\}_{(c,A,b)\in\mathcal{S}},$$

wobei u und y die unveränderlichen Variablen und v die Veränderlichen darstellen.

Wir wenden hier nicht den vollständigen Ansatz der adaptiven robusten Optimierung nach Ben-Tal u. a. [2] an, sondern passen ihn unserem Modell etwas an. Dies wollen wir jedoch erst in Kapitel 3 näher erläutern.

3 Robuste Terminvergabe unter Unsicherheiten

Terminvergaben und ihre Durchführbarkeit in Krankenhäusern werden von vielen Faktoren beeinflusst. Inbesondere kann man - wie bei vielen praktischen Anwendungen - nicht davon ausgehen, dass alle Daten und Behandlungen bei der Planung bekannt sind und berücksichtigt werden können. Dafür finden sich im Krankenhaus viele Beispiele, die diese Annahme belegen können, wie Notfallpatienten, deren Erscheinen vorher nicht bekannt gewesen ist, die aber schnell behandelt werden müssen, oder unvorhergesehen Diagnosen, auf deren Basis bisher nicht geplante Behandlungen durchgeführt werden müssen. Ein deterministischer Plan zur Terminvergabe, also ein Plan, der nur auf vorab bekannten Informationen basiert, ist in diesem Kontext in der Durchführung unpraktikabel, da er viele Ereignisse nicht beinhaltet und keine Möglichkeit bietet, auf Veränderungen einzugehen.

Eine Möglichkeit mit Unsicherheiten in der Terminvergabe umzugehen, ist ein robuster Terminplan, der alle möglichen Szenarien berücksichtigt und unter all diesen Szenarien das beste Ergebnis liefert. Die Gewinnung eines solchen robuste Plans zur Terminvergabe im Krankenhaus ist Hauptgegenstand dieser Arbeit. In diesem Kapitel werden zu dieser Problemstellung alle nötigen Informationen gegeben und erste Erkenntnisse ausgearbeitet.

Im Umfeld von Terminvergaben und -planungen im Krankenhausbereich ergeben sich viele mögliche Unsicherheiten, die jedoch nicht alle berücksichtigt werden sollen. Aus Zeitgründen beschränken wir uns in dieser Arbeit auf eine ausgewählte Unsicherheit und untersuchen diese genauer.

In Abschnitt 3.1 werden die Unsicherheiten der Terminvergabe im Krankenhaus erläutert. Die ausgewählte Unsicherheit der unsicheren klinischen Behandlungspfade und die dabei verwendeten Annahmen werden in Abschnitt 3.2 näher beschrieben. In Abschnitt 3.3 wird ein deterministische Modell der Terminvergabe in Krankenhäusern vorgestellt und im folgenden Abschnitt 3.4 ein robustes einstufiges sowie ein robustes zweistufiges Modell für unsichere Behandlungspfade erstellt und interpretiert.

3.1 Unsicherheiten bei der Terminvergabe in Krankenhäusern

Wie schon erwähnt, ist es insbesondere bei der Terminplanung in Krankenhäusern unwahrscheinlich, dass alle kommenden Patienten, Behandlungen und darüber hinausgehenden benötigten Daten zum Zeitpunkt der Planung vorliegen. Die Planer der Terminvergabe müssen also eine große Anzahl an Unsicherheiten und Veränderungen bei der Terminplanung einkalkulieren, hierzu zählen zum Beispiel stochastische Behandlungsdauern, eine Änderung des Behandlungspfads der Patienten, der Eingang vorher unbekannter Patienten, eine Änderung der Ressourcen, die für eine Behandlung benötigt werden, oder ein Maschinenausfall. Maschinen

können beispielsweise Röntgengeräte, aber auch Personal sein. Natürlich sind auch viele weitere Unsicherheitsfaktoren und vor allem Kombinationen von Unsicherheiten bei der Planung vorstellbar. Man sieht anhand dieser Beispiele schnell, dass ein deterministischer Terminplan, der jegliche Unsicherheiten unberücksichtigt lässt und keine Möglichkeit der Anpassung bietet, nicht realistisch ist und so zu hohen Kosten führen kann.

In dieser Masterarbeit werden *unsichere klinische Behandlungspfade* als einzige Unsicherheit gewählt und auf dieser Basis ein Lösungsverfahren für einen robusten Plan für die Terminvergabe in Krankenhäusern erarbeitet. Bis auf die Unsicherheiten der Behandlungspfade sind alle weiteren Daten bei der Planung bekannt und unveränderlich, insbesondere das Wissen über alle erscheinenden Patienten und die Behandlungsdauern. Ein klinischer Behandlungspfad gibt für einen Patienten alle Behandlungen und Untersuchungen an, die er während seines Krankenhausaufenthaltes erhalten wird. Ebenso liefert ein Behandlungspfad die Reihenfolge dieser Behandlungen untereinander. Im Kontext des Job Shop Scheduling Problems entsprechen Behandlungspfade der Reihenfolge von den Operationen eines Jobs.

3.2 Problemannahmen

Wir wollen nun die genauen Problemannahmen für diese Arbeit treffen. Allgemein nehmen wir an, dass in unserer Zeitplanung, die zum Beispiel einen Tag umfassen kann, n Patienten $J_1, \ldots, J_n$ erscheinen werden und m mögliche Behandlungsarten $M_1, \ldots, M_m$ zur Verfügung stehen. Die Indexmenge $\{1, \ldots, n\}$ aller Patienten bezeichnen wir im Folgenden mit I und die Indexmenge $\{1, \ldots, m\}$ aller Behandlungsarten mit J. Jede Behandlungsart umfasst genau eine Art der Behandlung oder Untersuchung, die einer bestimmten Ressourcenmenge zugeordnet ist. Zur Vereinfachung sind die Behandlungsarten alle unterschiedlich und jede genau einmal verfügbar. Außerdem wird für jede Behandlungsart eine bestimmte Ressourcenmenge, wie etwa Geräte oder Personal benötigt. Die Behandlungsart, zu der eine Behandlung O_{ij} für $i \in I$ und $j \in J$ gehört, wird mit $\mu_{ij} \in \{M_1 \ldots, M_m\}$ bezeichnet. Jeder Patient wird in einer bekannten und festgelegten Reihenfolge verschiedene Behandlungen erhalten, pro Patient J_i sind es n_i Behandlungen insgesamt mit $i \in I$. Eine bereits begonnene Behandlung eines Patienten darf nicht unterbrochen werden und es gibt keine Wechselzeiten auf den Behandlungsressourcen.

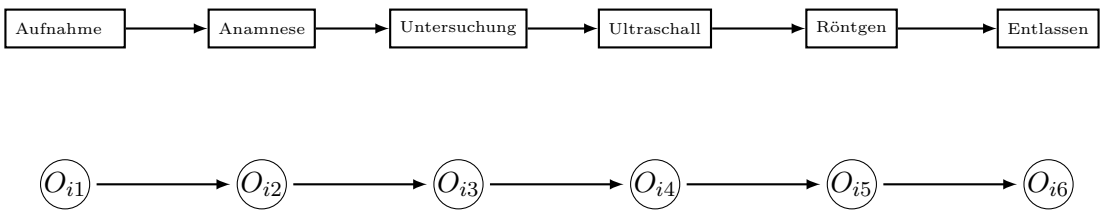


Abbildung 3.1: Behandlungspfad eines Patienten J_i

Abbildung 3.1 zeigt einen beispielhaften Behandlungspfad für einen möglichen Patienten J_i, $i \in I$. Dieser Patient durchläuft sechs Stationen im Krankenhaus, die Untersuchungen, Behandlungen oder verwaltungstechnischem Aufwand entsprechen. In einer festgelegten Reihenfolge wird der Patient J_i zunächst im Krankenhaus aufgenommen, bevor eine Anamnese durchgeführt wird. Nach einer medizinischen Untersuchung wird ein Ultraschall gemacht, bevor der Patient geröngt wird. Zuletzt wird der Patient wieder entlassen.

Anhand von Abbildung 3.1 lässt sich erkennen, dass man einen klinischen Behandlungspfad so interpretieren kann, dass jede mögliche Behandlung, Untersuchung oder sonstiges Ereignis des Behandlungspfads als Knoten eines Graphen dargestellt werden kann. Die Reihenfolge auf dem Behandlungspfad legt auch eine Reihenfolge zwischen den Knoten fest. Ein klinischer Behandlungspfad lässt sich somit als Graphenpfad interpretieren, bei dem jeder Knoten einer bestimmten Behandlung O_{ij} zuzuordnen ist mit $i \in I$ und $j \in J$. Im allgemeinen Fall entsprechen klinische Behandlungspfade keinem Graphenpfad, sondern eher einem Entscheidungsbaum, da bei Untersuchungen beispielsweise verschiedene Ausgänge möglich sind. In dieser Arbeit befassen wir uns jedoch ausschließlich mit einfachen Pfaden.

Wir nehmen an, dass für einen Patienten J_i mit $i \in I$ alle möglichen Behandlungen bekannt und die Reihenfolge der Behandlungen festgelegt ist. Im weiteren Verlauf dieser Arbeit bezeichnet eine Behandlung eines Patienten zur Vereinfachung sowohl die medizinischen Behandlungen, wie auch Untersuchungen oder andere Ereignisse im Krankenhaus. Einige dieser Behandlungen werden auf jeden Fall eintreten, bei anderen ist das tatsächliche Eintreten zum Planungszeitpunkt noch nicht bekannt, da sich während des Krankenhausaufenthaltes der Zustand eines Patienten verändern kann oder Diagnosen gemacht werden, die zusätzliche Behandlungen erfordern.

Für einen Patienten J_i bezeichnet O_i mit $|O_i| = n_i$ die gesamte Menge aller möglichen Behandlungen. $O_{i_{fix}} \subseteq O_i$ ist die Teilmenge der Behandlungen, die auf jeden Fall eintreffen werden, und die Teilmenge $O_{i_{var}} \subseteq O_i$ beinhaltet alle möglichen, zusätzlichen Behandlungen, deren Eintreffen vorher unsicher ist. Die Mengen $O_{i_{var}}$ und $O_{i_{fix}}$ sind disjunkt und es gilt, dass $O_{i_{var}} \cup O_{i_{fix}} = O_i$ ist.

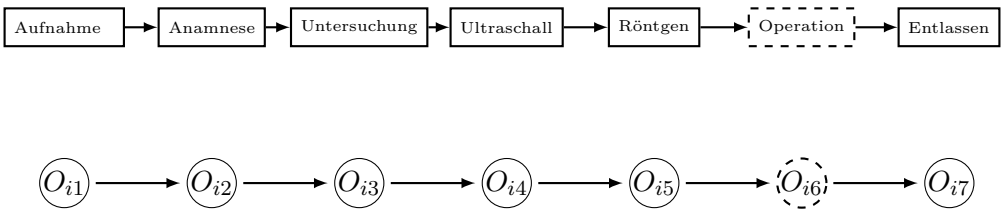


Abbildung 3.2: Unsicherer Behandlungspfad eines Patienten J

In Abbildung 3.2 lässt sich im Vergleich zu Abbildung 3.1 erkennen, wie ein möglicher unsicherer Behandlungspfad aussehen kann. Im Unterschied zu Abbildung 3.1 hat dieser Behand-

lungspfad sechs sicher eintretende Behandlungen, jedoch ist zum Planungszeitpunkt unklar, ob Behandlung O_{i6} tatsächlich eintreten wird. Für diesen Job J_i mit einer Behandlungsmenge O_i ist die Menge der fixen Behandlungen $O_{i_{fix}}$ die Menge aller Behandlungen ohne Behandlung O_{i6} und die Menge der möglicherweise zusätzlichen Behandlungen $O_{i_{var}}$ enthält somit nur die Behandlung O_{i6}.

Die unsicheren Behandlungen können über den gesamten Pfad verteilt und abwechselnd mit sicheren Behandlungen auftreten. Bekannt ist jedoch, welche zusätzlichen Behandlungen auf einem Behandlungspfad eintreten können, wo sie einzuordnen sind und wie lange diese Behandlungen dauern werden. Also sind explizit die zu einer Behandlung O_{ij} gehörende Behandlungsart $\mu_{ji} \in \{M_1, \ldots, M_m\}$, ihre Behandlungsdauer p_{ij} und ihre Einordnung auf den Behandlungspfaden für alle Patienten bekannt. Wir gehen bei der Modellierung dieser Problemstellung davon aus, dass alle einzuplanenden Patienten ab dem Planungsbeginn sofort verfügbar sind und keine weiteren Release- und Deadlinezeitpunkte für die Patienten vorliegen. Dies entspricht der Annahme, dass alle Patienten zum ersten Planungszeitpunkt im Krankenhaus sind und sofort mit der ersten Behandlung beginnen können.

Eine wichtige Annahme, die wir für diese Arbeit treffen, ist, dass im gesamten Planungszeitraum maximal K zusätzliche Behandlungen eintreten werden. Der Wert $K \geq 0$ ist dabei fest gegeben und ganzzahlig.

Wir wollen einen zulässigen Ablaufplan erzeugen, der die Summe aller letzten Behandlungszeitpunkte für alle Patienten minimiert. Dies entspricht der Zielfunktion

$$\min \sum_{i \in I} C_i.$$

C_i beschreibt hierbei den Endzeitpunkt der letzten Behandlung eines Patienten J_i, $i \in I$. Diese Zielfunktion trägt dazu bei, dass alle Patienten möglichst schnell behandelt werden.

Dass eine allgemeine deterministische Terminvergabe im Krankenhaus einer Instanz des Job Shop Scheduling Problems entspricht, lässt sich anhand der vorhergehenden Erläuterungen leicht nachvollziehen. Dabei entsprechen die Jobs den Patienten, die Operationen der Jobs den Behandlungen und Reihenfolgen werden über den Behandlungspfad geliefert. Die Prozessdauer einer Operation wird durch die Behandlungsdauer der dazugehörigen Behandlung bestimmt. Diese Problemstellung wird im Zuge der robusten Planung so erweitert, dass im Vorhinein nicht klar ist, welche K zusätzlichen Behandlungen eintreffen werden.

3.3 Deterministisches Modell

Auch wenn ein deterministischer Plan zur Koordination von Terminen in Krankenhäusern wenig praktikabel ist, wollen wir uns hier in diesen Abschnitt trotzdem kurz für ein tiefgehendes

Verständnis mit einem solchen befassen. Dazu verwenden wir die Annahmen und Notationen aus dem vorhergehenden Abschnitt 3.2. Es gibt hier jedoch keine Behandlungen, deren Eintreten vorher unsicher gewesen ist, weswegen $O_{i_{var}} = \emptyset$ für alle $i \in I$ gilt.

Die reelle Variable t_{ij} bezeichnet den Anfangszeitpunkt einer Behandlung O_{ij} mit $i \in I$ und $j \in J$ und die binäre Variable $x_{ij,i'j'}$ ist genau dann eins, wenn die Behandlungsarten der Behandlung O_{ij} von Patient J_i und der Behandlung $O_{i'j'}$ von Patient $J_{i'}$ gleich sind, also von der gleichen Behandlungsressource $\mu_{i'j'} = \mu_{ij}$ durchgeführt werden, und terminlich $O_{i'j'}$ direkt nach der Behandlung von O_{ij} durchgeführt wird. Anstelle eine Operation als O_{ij} zu schreiben, werden wir in dieser Arbeit häufig abkürzend ij für diese Behandlung notieren. R ist eine zusätzliche Dummybehandlung, die auf den letzten Auftrag jeder Behandlungsart folgen soll. Für diese Behandlung R gilt also $\mu_R = \{M_1, \ldots, M_m\}$.

Modell 2 (Deterministisches Modell)

$$\min \quad \sum_{i \in I} t_{in_i}$$

$$s.t. \quad \sum_{i' \in I \setminus \{i\}} \sum_{\substack{i'j' \in O_{i'} \cup \{R\}: \\ \mu_{ij} = \mu_{i'j'}}} x_{ij,i'j'} = 1 \qquad \forall ij \in O_i, i \in I \tag{2.1}$$

$$t_{ij} \geq t_{i(j-1)} + p_{i(j-1)} \qquad \forall ij, i(j-1) \in O_i, j > 1, i \in I \tag{2.2}$$

$$t_{ij} + p_{ij} \leq t_{i'j'} x_{ij,i'j'} \qquad \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I \tag{2.3}$$

$$x_{ij,i'j'} \in \{0,1\} \qquad \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I$$

$$t_{ij} \geq 0 \qquad \forall ij \in O_i, i \in I$$

Wir minimieren die Summe der Startzeitpunkte der letzten Behandlungen für jeden Patienten J_i, $i \in I$, anstelle in diesem deterministischen Modell 2 die Zielfunktion $\sum_{i \in I} C_i$ zu minimieren. Es gilt $C_i = t_{in_i} + p_{in_i}$ für $i \in I$, wobei p_{in_i} ein konstanter Wert ist. Für das Minimierungsproblem macht es also keinen Unterschied, über die in Modell 2 verwendete Zielfunktion oder über $\sum_{i \in I} C_i$ zu minimieren.

Ungleichung (2.1) fordert, dass jede Operation genau eine Nachfolgebehandlung eines weiteren Patienten besitzen muss. Diese Nachfolgebehandlung muss eine Behandlung der gleichen Behandlungsart sein, also laufen diese Behandlungen auf den gleichen Ressourcen ab. Mit der Ungleichung (2.2) wird eingehalten, dass eine Behandlung O_{ij} nur dann beginnen kann, wenn der Patient J_i seine vorausgehende Behandlung $O_{i(j-1)}$ abgeschlossen hat. Ungleichung (2.3) sagt in ähnlicher Weise aus, dass für eine Behandlungsart eine Patientenbehandlung nur beginnen kann, wenn die vorhergehende Patientenbehandlung für diese Behandlungsart ebenfalls abgeschlossen ist. Diese Vorgängerbeziehung wird über die Variable x definiert. Da sowohl x als auch t Variablen sind, ist die Ungleichung (2.3) somit quadratisch. Wir linearisieren diese,

indem wir mithilfe einer Konstante M die Ungleichung (2.3) durch die Ungleichung

$$t_{ij} + p_{ij} \leq t_{i'j'} + (1 - x_{ij,i'j'})M \qquad \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I \tag{2.3a}$$

in dem Modell 2 ersetzen. M ist hierbei ein ausreichend großer Wert, damit $t_{ij} + p_{ij} \leq t_{i'j'} + M$ für alle Behandlungen ij und $i'j'$ gilt.

Mithilfe von Modell 2 lässt sich also eine deterministische Terminvergabe im Krankenhaus bestimmen. Auf der Basis von diesem deterministischen Modell wird im nun folgenden Abschnitt ein robustes Modell unter unsicheren Behandlungspfaden eingeführt und erläutert.

3.4 Robuste Modellierungen der Terminvergabe unter unsicheren Behandlungspfaden

Wie schon erläutert, ist eine Lösung des deterministischen Modells aus Modell 2 nicht geeignet, um die komplexen Abläufe in Krankenhäusern zu koordinieren. Um einen guten Ablaufplan zu erzielen, muss man alle möglichen Zustände, die eintreten könnten, berücksichtigen. Diese Zustände - oder auch Szenarien - sind in unserem Fall alle Möglichkeiten aus allen zusätzlichen Behandlungen, deren Eintreten unsicher ist, K Stück auszuwählen.

Wir werden sowohl ein einstufiges robustes Optimierungsmodell wie auch ein zweistufiges robustes Optimierungsmodell betrachten. Diese unterscheiden sich in der Flexibilität, die ein Terminplan besitzt, der mit diesen Modellen erstellt wurde.

3.4.1 Einstufige robuste Optimierung

Zunächst wollen wir ein einstufiges robustes Optimierungsmodell betrachten. Dabei werden alle Entscheidungsvariablen vor Eintreten des Szenarios festgelegt und danach nicht mehr verändert.

Für einen robusten Terminplan suchen wir eine robust optimale Lösung (x, t), für die gilt, dass in allen Szenarien die Nebenbedingungen eingehalten werden und für die der Worst-Case-Wert $\hat{c}(x,t) = \max\limits_{(c,A,b)\in\mathcal{S}} c^T(x,t)$ minimal ist.

Zunächst definieren wir den robusten Counterpart als robustes einstufiges Modell ausgehend von dem deterministischen Modell 2. Sei dazu O_i die Menge der deterministischen Behandlungen eines Patienten J_i. Wir nehmen an, dass nur die Behandlungspfade unsicher sind, das bedeutet, dass für ein Szenario S nur die Menge der Behandlungen O_i von dem Szenario abhängt. Dafür schreiben wir zunächst O_i^S für $S \in \mathcal{S}$.

Modell 3 (Robuster Counterpart)

$$
\begin{aligned}
\min \quad & y \\
s.t. \quad & \sum_{i' \in I \setminus \{i\}} \sum_{\substack{i'j' \in O_{i'}^S \cup \{R\}: \\ \mu_{ij} = \mu_{i'j'}}} x_{ij,i'j'} = 1 && \forall ij \in O_i^S, i \in I, S \in \mathcal{S} && (3.1)\\
& t_{ij} \geq t_{i(j-1)} + p_{i(j-1)} && \forall ij, i(j-1) \in O_i^S, j > 1, i \in I, S \in \mathcal{S} && (3.2)\\
& t_{ij} + p_{ij} \leq t_{i'j'} + (1 - x_{ij,i'j'})M && \forall ij \in O_i^S, i'j' \in O_{i'}^S, i, i' \in I, S \in \mathcal{S} && (3.3)\\
& \sum_{i \in I} t_{in_i} \leq y && && (3.4)\\
& x_{ij,i'j'} \in \{0,1\} && \forall ij \in O_i^S, i'j' \in O_{i'}^S, i, i' \in I, S \in \mathcal{S} &&\\
& t_{ij} \geq 0 && \forall ij \in O_i^S, i \in I, S \in \mathcal{S} &&
\end{aligned}
$$

Wir wollen jedoch, um die Szenarien besser umfassen zu können, diesen robusten Counterpart umformulieren. Wir suchen eine robust zulässige Lösung (x, t), die den besten Worst-Case-Wert besitzt. Damit die Lösung aber in allen Szenarien eine gültige ist, benötigen wir Reihenfolgen für die einzelnen Behandlungsarten, die alle möglichen zusätzlichen Behandlungen der Patienten einschließen. Dazu bestimmen wir für alle Behandlungen, egal, ob sie tatsächlich eintreten werden oder nicht, eine Reihenfolge auf den Ressourcen der dazugehörigen Behandlungsart. Außerdem bestimmen wir für jede Behandlung O_{ij} einen Startzeitpunkt t_{ij}. Unsere Szenarien bestehen darin, dass K zusätzliche Behandlungen maximal eintreten werden. Das folgende Modell 4 beschreibt unser somit betrachtetes robuste Modell. Dieses nennen wir zukünftig das robuste einstufige Modell.

Modell 4 (Robustes einstufiges Modell)

$$
\begin{aligned}
\min_{x,t} \max_{z} & \sum_{i \in I} t_{in_i} \\
s.t. \quad & \sum_{i' \in I \setminus \{i\}} \sum_{\substack{i'j' \in O_{i'} \cup \{R\} \\ \mu_{ij} = \mu_{i'j'}}} x_{ij,i'j'} = 1 && \forall \quad ij \in O_i, i \in I && (4.1)\\
& t_{ij} \geq t_{i(j-1)} + p_{i(j-1)} z_{i(j-1)} && \forall ij, i(j-1) \in O_i, j > 1, i \in I && (4.2)\\
& t_{ij} + p_{ij} z_{ij} \leq t_{i'j'} + (1 - x_{ij,i'j'})M && \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I && (4.3)\\
& z_{ij} = 1 && \forall ij \in O_{i_{fix}}, i \in I && (4.4)\\
& \sum_{i \in I} \sum_{ij \in O_{i_{var}}} z_{ij} \leq K && && (4.5)\\
& x_{ij,i'j'} \in \{0,1\} && \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I &&\\
& t_{ij} \geq 0 && \forall ij \in O_i, i \in I &&\\
& z_{ij} \in \{0,1\} && \forall ij \in O_i, \forall i \in I &&
\end{aligned}
$$

Da wir Startzeitpunkte und Reihenfolgen für alle Behandlungen berechnet haben, hängt nur die Behandlungsdauer davon ab, ob die Behandlung tatsächlich eintritt oder nicht. Im Falle, dass eine zusätzliche Behandlung eintritt, erhält sie eine Behandlungsdauer, die der tatsächlichen Behandlungsdauer entspricht. Für den Fall, dass diese Behandlung nicht eintritt, erhält sie eine Behandlungsdauer von 0. Die Szenarien beschreiben wir in Modell 4 mit einer binären Variable z. Falls eine Behandlung O_{ij} eintritt, ist $z_{ij} = 1$ und 0 sonst. Dieses robuste Optimierungsmodell entspricht dem robusten Counterpart unserer Problemstellung, jedoch mit einer genaueren Darstellung der Szenarien.

Für das Job Shop Scheduling bedeutet dies, dass alle Behandlungen und damit alle Knoten des disjunktiven Graphen fest sind, jedoch die Kosten der Knoten des disjunktiven Graphen unsicher sind. Für fest eintretende Behandlungen, entsprechen die Kosten weiterhin der Behandlungsdauer. Bei einer nicht sicher eintretenden Behandlung sind die Kosten entweder 0 oder entsprechen der Behandlungsdauer, wobei nur K Behandlungen, die nicht sicher eintreten werden, mit Kosten der Behandlungsdauer gewählt werden dürfen.

Das Modell 4 liefert, wie der robuste Counterpart, die konservativste robuste Lösung, da alle Entscheidungen vor Bekanntwerden des tatsächlichen Szenarios getroffen werden.
Die Nebenbedingung (4.1) entspricht der exakten Nebenbedingung (2.1) des deterministischen Modells. Mit diesen beiden Nebenbedingungen werden die Patientenreihenfolgen für die Behandlungsarten bestimmt. Nebenbedingung (4.4) besagt, dass die Variable z für die definitiv eintretenden Behandlungen gleich 1 ist. Die Nebenbedingungen (4.2) und (4.3) sind die entsprechenden unsicheren Nebenbedingungen der deterministischen Ungleichungen (2.2) und (2.3), die aussagen, dass eine Behandlung nur nach Beendigung der vorhergehenden Patientenbehandlung und der Vorgängerbehandlung der Behandlungsart erfolgen darf. Egal, ob die Behandlung eintreten wird oder nicht, wird für sie ein Startzeitpunkt berechnet, nur die Behandlungsdauern hängen davon ab, ob die Behandlung eintritt oder nicht. Dies wird im Unterschied zu dem deterministischen Modell 2 so betrachtet, dass die Prozesszeiten mit der Variable z multipliziert werden.
Anhand dieser Ungleichungen sieht man, dass sich ein mögliches Szenario in unserer gewählten Modellierung nur auf die Behandlungsdauern bezieht.

In allen möglichen robust zulässigen Lösungen (x, t) wird nun diejenige gesucht, die für das schlechteste Szenario die beste Lösung besitzt, also den besten Worst-Case-Wert.

Diese erste robuste Formulierung 4 kann zu einem linearen Modell umgewandelt werden, indem es wie der robuste Counterpart zu einem reinen Minimierungsproblem umgewandelt wird. Das innere Problem Stufe von Modell 4, das Maximierungsproblem des Max-Szenario-Problems mit der Zielfunktion $\max\limits_{z} \sum\limits_{i \in I} t_{in_i}$, wird dabei als Nebenbedingung $v \geq \sum\limits_{i \in I} t_{in_i}$ in dem äußeren Modell formuliert. Anschließend wird über v minimiert. So ergibt sich unser lineares robustes einstufiges Optimierungsmodell 5.

Modell 5 (Robustes einstufiges Modell)

$$
\begin{aligned}
\min \quad & v \\
s.t. \quad & \sum_{i' \in I \setminus \{i\}} \sum_{\substack{i'j' \in O_{i'} \cup \{R\} \\ \mu_{ij} = \mu_{i'j'}}} x_{ij,i'j'} = 1 && \forall ij \in O_i, i \in I && (5.1) \\
& v \geq \sum_{i \in I} t_{in_i} && && (5.2) \\
& t_{ij} \geq t_{i(j-1)} + p_{i(j-1)} z_{i(j-1)} && \forall ij, i(j-1) \in O_i, j > 1, i \in I && (5.3) \\
& t_{ij} + p_{ij} z_{ij} \leq t_{i'j'} + (1 - x_{ij,i'j'}) M && \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I && (5.4) \\
& z_{ij} = 1 && \forall ij \in O_{i_{fix}}, i \in I && (5.5) \\
& \sum_{i \in I} \sum_{ij \in O_{i_{var}}} z_{ij} \leq K && && (5.6) \\
& x_{ij,i'j'} \in \{0,1\} && \forall ij \in O_i, (i'j') \in O_{i'}, i, i' \in I && \\
& t_{ij} \geq 0 && \forall ij \in O_i, i \in I && \\
& z_{ij} \in \{0,1\} && \forall ij \in O_i, i \in I &&
\end{aligned}
$$

Der soeben vorgestellte Modellansatz ist ein sehr konservatives robustes Optimierungsmodell, da die Lösung für alle Szenarien zulässig sein muss. Um mehr Flexibilität zu erlangen, betrachten wir nun eine Idee der adaptiven robusten Optimierung, nämlich ein zweistufiges robustes Optimierungsproblem.

3.4.2 Zweistufige robuste Optimierung

Bei der einstufigen robusten Optimierung nach Modell 5 wird eine Lösung (x, t) berechnet, die für alle Szenarien und somit auch im schlechtesten Fall das beste Ergebnis liefert. Die Festlegung beider Entscheidungsvariablen geschieht dabei vor Bekanntwerden des tatsächlichen Szenarios. Bei dem Ansatz der adaptiven robusten Optimierung müssen nicht alle Entscheidungsvariablen vor Eintreten eines Szenarios festgesetzt werden. Die Variablen werden aufgeteilt in unveränderliche Variablen, für die es unerlässlich ist, dass sie vor Eintreten des Szenarios bestimmt werden, und veränderliche Variablen, die nach Eintreten des Szenarios bestimmt werden dürfen. Dadurch kann man nach Kenntnis des tatsächlichen Szenarios flexibler auf die Problemstellung eingehen.

Um diese Idee auf unser Problem zu übertragen, betrachten wir noch einmal das einstufige robuste Optimierungsmodell 4. Wir benötigen eine robust zulässige Lösung, die für alle möglichen Szenarien zulässig ist. Da zum Planungszeitpunkt unklar ist, welche Behandlungen tatsächlich stattfinden werden, müssen alle Patientenreihenfolgen für alle Behandlungsarten vor Kenntnis über das Szenario geplant werden. Also entspricht x einer unveränderlichen Variablen. Es ist jedoch nicht zwingend nötig, die Startzeitpunkte der Behandlungen vor dem Eintreten des

Szenarios zu bestimmen. Hat man die Patientenreihenfolgen für die Behandlungsarten gegeben und weiß man, welche Behandlungen tatsächlich eintreten werden, können daraufhin die optimalen Startzeiten berechnet werden. Es ist also sinnvoll, die Startzeitpunkte in Abhängigkeit der Szenarien zu bestimmen. Dazu seien die Startzeitpunkte t veränderliche Variablen.

Das Modell 4 ändert sich für diesen flexibleren Ansatz nun so, dass die Entscheidungsvariablen x und t nicht mehr zusammen vor dem Eintreten des Szenarios bestimmt werden, sondern die Startzeiten erst nach Wissen über das Szenario. Dadurch ergibt sich das folgende zweistufige Optimierungsmodell.

Modell 6 (Zweistufiges robustes Modell 1)

$$\min_{x} \max_{z} \min_{t} \quad \sum_{i \in I} t_{in_i}$$

$$s.t. \quad \sum_{i' \in I \setminus \{i\}} \sum_{\substack{i'j' \in O_{i'} \cup \{R\} \\ \mu_{ij} = \mu_{i'j'}}} x_{ij,i'j'} = 1 \qquad \forall ij \in O_i, i \in I \tag{6.1}$$

$$t_{ij} \geq t_{i(j-1)} + p_{i(j-1)} z_{i(j-1)} \qquad \forall ij, i(j-1) \in O_i, j > 1, i \in I \tag{6.2}$$

$$t_{ij} + p_{ij} z_{ij} \leq t_{i'j'} + (1 - x_{ij,i'j'}) M \qquad \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I \tag{6.3}$$

$$z_{ij} = 1 \qquad \forall ij \in O_{i_{fix}}, i \in I \tag{6.4}$$

$$\sum_{i \in I} \sum_{(ij) \in O_{i_{var}}} z_{ij} \leq K \tag{6.5}$$

$$x_{ij,i'j'} \in \{0,1\} \qquad \forall ij \in O_i, (i'j') \in O_{i'}, i, i' \in I$$

$$t_{ij} \geq 0 \qquad \forall ij \in O_i, i \in I$$

$$z_{ij} \in \{0,1\} \qquad \forall ij \in O_i, i \in I$$

Im Vergleich zu Modell 5 ist aus dem einstufigen Entscheidungsmodell mit einer Zielfunktion nun ein zweistufiges Modell mit drei Zielrichtungen geworden. Geändert hat sich jedoch im Vergleich zu dem zu Modell 5 äquivalenten Modell 4 nur die Zeitpunkte der Berechnungen der Variablen. Im einstufigen robusten Modell 5 wurden die Entscheidungen x und t vor dem Bekanntwerden des Szenarios auf Grundlage des schlechtmöglichsten Szenarios getroffen. Im zweistufigen robusten Modell treffen wir nur die Reihenfolge x auf der Basis eines optimalen Zeitplans für das schlechteste Szenario. Dabei wird der Worst-Case-Wert nicht mehr in Abhängigkeit von x und t bestimmt, sondern der Worst-Case-Wert beinhaltet nun zusätzlich das Problem der Terminvergabe der Szenarien.

Kompaktere Formulierungen

Das Modell 6 besitzt zunächst drei Zielfunktionen. Dies wollen wir versuchen, etwas kompakter zu formulieren. Setzen wir die Entscheidung der ersten Stufe als bereits getätigt voraus, verbleibt noch das Problem, über alle möglichen Szenarien den Worst-Case-Wert zu finden. Der

Worst-Case-Wert wird durch das Szenario definiert, dessen optimaler Zeitplan der schlechteste aller Szenarien ist. Der Zeitplan wird dabei für alle Szenarien als deterministischer Plan zur Terminvergabe berechnet, ohne Reihenfolgen zu berechnen. Diese Worst-Case-Stufe umfasst aktuell zwei unterschiedliche Zielrichtungen. Wie zuvor bei dem einstufigen robusten Optimierungsmodell lässt sich dies jedoch als Modell mit nur einer Zielrichtung umformulieren. Dazu wird die innerste Zielfunktion

$$\min \sum_{i \in I} t_{in_i}$$

für ein fest gewähltes x als lineare Nebenbedingung

$$v(x) \leq \sum_{i \in I} t_{in_i}$$

in das Modell eingebaut, womit sich das zweistufige Modell mit verbleibenden zwei Zielrichtungen ergibt.

Modell 7 (Zweistufiges robustes Modell 2)

$$\begin{aligned}
\min_{x} \quad \max_{z,t,v} \quad & v(x) \\
s.t. \quad & \sum_{i' \in I \setminus \{i\}} \sum_{\substack{i'j' \in O_{i'} \cup \{R\} \\ \mu_{ij} = \mu_{i'j'}}} x_{ij,i'j'} = 1 && \forall ij \in O_i, i \in I && (7.1) \\
& v(x) \leq \sum_{i \in I} t_{in_i} && && (7.2) \\
& t_{ij} \geq t_{i(j-1)} + p_{i(j-1)} z_{i(j-1)} && \forall ij, i(j-1) \in O_i, j > 1, i \in I && (7.3) \\
& t_{ij} + p_{ij} z_{ij} \leq t_{i'j'} + (1 - x_{ij,i'j'}) M && \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I && (7.4) \\
& z_{ij} = 1 && \forall ij \in O_{i_{fix}}, i \in I && (7.5) \\
& \sum_{i \in I} \sum_{ij \in O_{i_{var}}} z_{ij} \leq K && && (7.6) \\
& x_{ij,i'j'} \in \{0,1\} && \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I && \\
& t_{ij} \geq 0 && \forall ij \in O_i, i \in I && \\
& z_{ij} \in \{0,1\} && \forall ij \in O_i, i \in I &&
\end{aligned}$$

Es verbleiben aber auch für diese zweite Formulierung des zweistufigen robusten Optimierungsmodells zwei Zielrichtungen. Für eine kompakte Darstellung dieses zweistufigen robusten Optimierungsmodells eignet sich das vorherige Vorgehen nun nicht mehr. Hierfür müssten wir die Zielfunktion

$$\max \quad v(x)$$

durch eine Nebenbedingung

$$w \geq v(x)$$

für alle möglichen Reihenfolgen x ausdrücken und w minimieren. Dies entspräche dem folgenden Modell:

Modell 8

$$\begin{aligned}
\min \quad & w \\
s.t. \quad & \sum_{i' \in I \setminus \{i\}} \sum_{\substack{i'j' \in O_{i'} \cup \{R\} \\ \mu_{ij} = \mu_{i'j'}}} x_{ij,i'j'} = 1 && \forall ij \in O_i, i \in I && (8.1) \\
& v(x) \leq \sum_{i \in I} t_{in_i} && && (8.2) \\
& w \geq v(x) && \forall x && (8.3) \\
& t_{ij} \geq t_{i(j-1)} + p_{i(j-1)} z_{i(j-1)} && \forall ij, i(j-1) \in O_i, j > 1, i \in I && (8.4) \\
& t_{ij} + p_{ij} z_{ij} \leq t_{i'j'} + (1 - x_{ij,i'j'}) M && \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I && (8.5) \\
& z_{ij} = 1 && \forall ij \in O_{i_{fix}}, i \in I && (8.6) \\
& \sum_{i \in I} \sum_{ij \in O_{ivar}} z_{ij} \leq K && && (8.7) \\
& x_{ij,i'j'} \in \{0,1\} && \forall ij \in O_i, (i'j') \in O_{i'}, i, i' \in I && \\
& t_{ij} \geq 0 && \forall ij \in O_i, i \in I && \\
& z_{ij} \in \{0,1\} && \forall ij \in O_i, i \in I &&
\end{aligned}$$

Damit beide Formulierungen der Modelle 7 und 8 äquivalent sind, müssen die jeweiligen optimalen Lösungen den gleichen Wert ergeben. Das Modell 8 ist jedoch unbeschränkt, da für v und w keine weiteren Schranken vorliegen und die Nebenbedingungen (8.2) und (8.3) sowie die Zielfunktion dies bedingen. Die Variable w soll minimiert werden und w ist größer als $v(x)$ für alle möglichen Patientenreihenfolgen x für die Behandlungsarten. Da jedoch an jedes $v(x)$ nur eine obere Schranke gegeben ist, ist die Variable w somit nicht nach unten beschränkt. Also kann dieses Modell nicht der Formulierung 7 entsprechen.

Ein anderer Betrachtungsansatz für eine kompaktere Formulierung

Der bisher betrachtete Ansatz, um das Modell 4 in eine kompakter Form zu bringen, führte zu Modell 7 mit zwei Zielrichtungen. Wir sehen in diesem Abschnitt nun eine weiteren Ansatz, eine kompaktere Form von Modell 4 zu erhalten. Diese beinhaltet, wie wir in Abschnitt 3.4.3 sehen werden, ein graphentheoretisches Problem, was wir näher untersuchen wollen und mit dessen

Hilfe wir die robuste Terminvergabe im Krankenhaus berechnen wollen. Um diese kompaktere Form zu erhalten, gehen wir noch einmal zurück zu unserem Ausgangsmodell 4. Wir nehmen an, dass die Reihenfolgen x bereits festgelegt sind und ein Szenario S, über die Variablen z definiert, eingetroffen ist. Es verbleibt das Optimierungsmodell der zweiten Entscheidungsstufe.

Modell 9 (Zweite Entscheidungsstufe)

$$\begin{aligned}
\min \quad & \sum_{i \in I} t_{in_i} & & & & \\
s.t. \quad & t_{ij} \geq t_{i(j-1)} + p_{i(j-1)} z_{i(j-1)} & & \forall ij, i(j-1) \in O_i, j > 1, i \in I & [\pi] \quad & (9.1) \\
& t_{ij} + p_{ij} z_{ij} \leq t_{i'j'} + (1 - x_{ij,i'j'}) M & & \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I & [y] \quad & (9.2) \\
& t_{ij} \geq 0 & & \forall ij \in O_i, i \in I & &
\end{aligned}$$

Diese zweite Entscheidungsstufe beinhaltet nur die Festlegung der genauen Starttermine anhand der bereits festgelegten Reihenfolge x und einem eingetretenen Szenario S. Die Werte x und z sind damit genau wie die Prozesszeiten p als Konstanten zu werten, da diese Entscheidungen schon getroffen wurden. Die Werte von x geben an, welche Reihenfolgen für die Behandlungsarten einzuhalten sind, und die Werte von z geben an, welche Behandlungen eingetreten sind. Da insbesondere die Variable x fest gesetzt ist, können wir auch die eigentliche quadratische Nebenbedingung

$$t_{ij} + p_{ij} z_{ij} \leq t_{i'j'} x_{ij,i'j'} \qquad \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I$$

anstelle von Nebenbedingung (9.2) betrachten, da sie hier nun linear ist. Das Modell 9 ist ein lineares Programm mit folgender zugrundeliegender Koeffizientenmatrix der Nebenbedingungen:

$$A = \left(\begin{array}{cccccc}
-1 & 1 & & & & \\
 & -1 & 1 & & & \\
 & & & \ddots & & \\
 & & & & -1 & 1 \\
\hline
-1 & x_{ij,i'j'} & & & & \\
 & & & \vdots & & \\
-1 & & & & & x_{ij,i'j'} \\
 & & & \vdots & & \\
x_{ij,i'j'} & & & & & -1 \\
 & & & \vdots & & \\
 & & & & x_{ij,i'j'} & -1
\end{array}\right)$$

wobei die Werte x binär sind und für eine Behandlung O_{ij} maximal ein Wert $x_{ij,i'j'}$ für alle übrigen Behandlungen $O_{i'j'}$ gleich eins ist. Also ist die transponierte Matrix A^T als Teilmatrix einer Inzidenzmatrix eines Digraphen total unimodular. Somit ist eine optimale Lösung von

Modell 9 immer ganzzahlige. Diese Ganzzahligkeitseigenschaft bleibt auch bei der Dualisierung erhalten. Dualisiert man dieses Modell 9 mithilfe der Dualvariablen π und y, so erhält man das folgende Modell:

Modell 10 (Duales Modell der zweiten Entscheidungsstufe)

$$\max \quad \sum_{i\in I}\Big(\sum_{ij\in O_i}\big[p_{i(j-1)}z_{i(j-1)}\pi_{ij} + \sum_{i'\in I\setminus\{i\}}\Big(\sum_{i'j'\in O_{i'}} p_{ij}z_{ij}y_{ij,i'j'}\Big)\big]\Big)$$

$$s.t. \quad -\pi_{ij} + \pi_{i(j-1)} - \sum_{i'\in I}\Big(\sum_{i'j'\in O_{i'}} y_{ij,i'j'} - x_{i'j',ij}y_{i'j',ij}\Big) \le 0 \qquad \forall ij, i(j-1) \in O_i\setminus\{in_i\}, i \in I \tag{10.1}$$

$$\pi_{i(n_i-1)} + \sum_{i'\in I}\Big(\sum_{i'j'\in O_{i'}} x_{i'j',in_i}y_{i'j',in_i}\Big) \le 1 \qquad \forall i \in I \tag{10.2}$$

$$\pi_{ij} \ge 0 \qquad \forall ij \in O_i, i \in I$$

$$y_{ij,i'j'} \ge 0 \qquad \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I$$

Auch die Matrix, die dem Polyeder der Formulierung von Modell 10 zugrunde liegt, ist total unimodular und daher nimmt auch das duale Programm der zweiten Entscheidungsstufe immer eine ganzzahlige Lösung an. Somit gelten für eine optimale Lösung von Modell 10 die Ganzzahligkeitsbedingungen

$$y \in \mathbb{Z}^+, \pi \in \mathbb{Z}^+.$$

Auf nähere Erläuterungen und Interpretationen zu Modell 10 wollen wir erst in Abschnitt 3.4.3 eingehen. Sowohl das Max-Szenario-Problem als zweite Zielfunktion wie auch dieses dualisierte zweite Entscheidungsproblem sind nun Maximierungsprobleme. Wir können sie daher in einem gemeinsamen Modell zusammenfassen. Dieses bezeichnen wir als die Max-Szenario-Formulierung.

Modell 11 (Max-Szenario-Formulierung)

$$\max \quad \sum_{i\in I}\Big(\sum_{ij\in O_i}\big[p_{i(j-1)}z_{i(j-1)}\pi_{ij} + \sum_{i'\in I\setminus\{i\}}\Big(\sum_{i'j'\in O_{i'}} p_{ij}z_{ij}y_{ij,i'j'}\Big)\big]$$

$$s.t. \quad -\pi_{ij} + \pi_{i(j-1)} - \sum_{i'\in I}\Big(\sum_{i'j'\in O_{i'}} y_{ij,i'j'} - x_{i'j',ij}y_{i'j',ij}\Big) \le 0 \qquad \forall ij, i(j-1) \in O_i\setminus\{in_i\}, i \in I \tag{11.1}$$

$$\pi_{i(n_i-1)} + \sum_{i'\in I}\Big(\sum_{i'j'\in O_{i'}} x_{i'j',in_i}y_{i'j',in_i}\Big) \le 1 \qquad \forall i \in I \tag{11.2}$$

$$z_{ij} = 1 \qquad \forall ij \in O_{i_{fix}}, i \in I \tag{11.3}$$

$$\sum_{i\in I}\sum_{ij\in O_{i_{var}}} z_{ij} \le K \tag{11.4}$$

$$\pi_{ij} \ge 0 \qquad \forall ij \in O_i, i \in I$$

$$y_{ij,i'j'} \ge 0 \qquad \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I$$

$$z_{ij} \in \{0,1\} \qquad \forall ij \in O_i, i \in I$$

Die Matrix A der Nebenbedingungen, die dieser Formulierung zugrunde liegt,

$$A = \left(\begin{array}{cccccccc}
-1 & 1 & & & & & & \\
 & -1 & 1 & & & & & \\
 & & \ddots & & & & & \\
 & & & -1 & 1 & & & \\
\hline
-1 & x_{ij,i'j'} & & & & & & \\
 & & \vdots & & & & & \\
-1 & & & & x_{ij,i'j'} & & & \\
 & & \vdots & & & & & \\
x_{ij,i'j'} & & & & -1 & & & \\
 & & \vdots & & & & & \\
 & & & x_{ij,i'j'} & -1 & & & \\
\hline
 & & & & & 1 & 1 \ \ldots \ 1 & \\
\hline
 & & & & & & & 1 \\
 & & & & & & 1 & \\
 & & & & & \ddots & & \\
 & & & & 1 & & &
\end{array}\right)$$

ist zwar ebenfalls total unimodular, jedoch besitzt das Modell 11 keine lineare Zielfunktion. Aus diesen Gründen kann der Dualitätssatz hier kein zweites Mal angewendet werden, um eine kompakte Formulierung für das zweistufige robuste Optimierungsproblem zu erhalten. Wie wir später in Abschnitt 3.4.3 noch sehen werden, ändert eine Linearisierung der Zielfunktion auch die Nebenbedingungsmatrix A, die bei erfolgreicher Linearisierung der Zielfunktion ihre totale Unimodularität verliert. Dadurch verliert das Modell seine Ganzzahligkeitsbedingung. Da z binär ist, kann also der Dualitätssatz nicht angewandt werden. Bisher ergibt das Modell 11 eingebettet in die erste Entscheidungsstufe das weiterhin zweistufige Entscheidungsmodell mit zwei Zielrichtungen:

Modell 12 (Zweistufiges robustes Modell 3)

$$\min_{x} \max_{z,y,\pi} \quad \sum_{i\in I}\Big(\sum_{ij\in O_i}\big[p_{i(j-1)}z_{i(j-1)}\pi_{ij} + \sum_{i'\in I\setminus\{i\}}\Big(\sum_{i'j'\in O_{i'}} p_{ij}z_{ij}y_{ij,i'j'}\Big)\big]$$

$$\begin{aligned}
s.t. \quad & \sum_{i'\in I\setminus\{i\}}\sum_{\substack{i'j'\in O_{i'}\cup\{R\}\\ \mu_{ij}=\mu_{i'j'}}} x_{ij,i'j'} = 1 && \forall ij\in O_i, i\in I && (12.1)\\
& -\pi_{ij}+\pi_{i(j-1)} - \sum_{i'\in I}\Big(\sum_{i'j'\in O_{i'}} y_{ij,i'j'} - x_{i'j',ij}y_{i'j',ij}\Big) \le 0 && \forall ij, i(j-1)\in O_i\setminus\{in_i\}, i\in I && (12.2)\\
& \pi_{i(n_i-1)} + \sum_{i'\in I}\Big(\sum_{i'j'\in O_{i'}} x_{i'j',in_i}y_{i'j',in_i}\Big) \le 1 && \forall i\in I && (12.3)\\
& z_{ij} = 1 && \forall ij\in O_{i_{fix}}, i\in I && (12.4)\\
& \sum_{i\in I}\sum_{ij\in O_{i_{var}}} z_{ij} \le K && && (12.5)\\
& x_{ij,i'j'}\in\{0,1\} && \forall ij\in O_i, i'j'\in O_{i'}, i,i'\in I\\
& \pi_{ij}\ge 0 && \forall ij\in O_i, i\in I\\
& y_{ij,i'j'}\ge 0 && \forall ij\in O_i, i'j'\in O_{i'}, i,i'\in I\\
& z_{ij}\in\{0,1\} && \forall ij\in O_i, i\in I
\end{aligned}$$

Dieses zweistufige Modell 12 mit den verbleibenden zwei Zielrichtungen lässt sich, wie wir schon mehrfach in dieser Arbeit gesehen haben, durch ein Ersetzen der Zielfunktion

$$\max \sum_{i\in I}\Big(\sum_{ij\in O_i}\big[p_{i(j-1)}z_{i(j-1)}\pi_{ij} + \sum_{i'\in I\setminus\{i\}}\Big(\sum_{i'j'\in O_{i'}} p_{ij}z_{ij}y_{ij,i'j'}\Big)\big]$$

durch die Nebenbedingung

$$v \ge \sum_{i\in I}\Big(\sum_{ij\in O_i}\big[p_{i(j-1)}z_{i(j-1)}\pi_{ij} + \sum_{i'\in I\setminus\{i\}}\Big(\sum_{i'j'\in O_{i'}} p_{ij}z_{ij}y_{ij,i'j'}\Big)\big]$$

in ein kompaktes Gesamtmodell umformulieren.

Modell 13 (Kompakte Formulierung)

$$\min \quad v$$

$$s.t. -\pi_{ij} + \pi_{i(j-1)} - \sum_{i' \in I} \Big(\sum_{i'j' \in O_{i'}} y_{ij,i'j'} + x_{i'j',ij} y_{i'j',ij} \Big) \leq 0 \qquad \forall ij, i(j-1) \in O_i \backslash \{in_i\}, i \in I \tag{13.1}$$

$$\pi_{i(n_i-1)} + \sum_{i' \in I} \Big(\sum_{i'j' \in O_{i'}} x_{i'j',in_i} y_{i'j',in_i} \Big) \leq 1 \qquad \forall i \in I \tag{13.2}$$

$$z_{ij} = 1 \qquad \forall ij \in O_{i_{fix}}, i \in I \tag{13.3}$$

$$\sum_{i \in I} \sum_{ij \in O_{i_{var}}} z_{ij} \leq K \tag{13.4}$$

$$v \geq \sum_{i \in I} \Big(\sum_{ij \in O_i} [p_{i(j-1)} z_{i(j-1)} \pi_{ij} + \sum_{i' \in I \backslash \{i\}} \Big(\sum_{i'j' \in O_{i'}} p_{ij} z_{ij} y_{ij,i'j'} \Big)] \tag{13.5}$$

$$x_{ij,i'j'} \in \{0,1\} \qquad \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I$$

$$\pi_{ij} \geq 0 \qquad \forall ij \in O_i, i \in I$$

$$y_{ij,i'j'} \geq 0 \qquad \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I$$

$$z_{ij} \in \{0,1\} \qquad \forall ij \in O_i, i \in I$$

$$v \geq 0$$

Dies ist zwar eine kompakte Formulierung des Gesamtmodells, jedoch sind einige Nebenbedingungen quadratisch und somit beschreiben diese Nebenbedingungen kein Polyeder mehr. Dieses Problem gehört der konvexen Optimierung an. Methoden, ein solches Modell zu lösen sind zum Beispiel Subgradientenverfahren oder die innere Punkt Methoden. Diese wollen wir jedoch in dieser Arbeit nicht näher betrachten. Wir kehren somit an dieser Stelle zu der Betrachtung des Max-Szenario-Modell 11 zurück, da dies noch nicht weiter untersucht und beschrieben worden ist.

3.4.3 Interpretation der Max-Szenario-Formulierung

Unser im vorherigen Abschnitt definiertes Max-Szenario-Modell 11,

Modell 14

$$\max \sum_{i\in I}\Big(\sum_{ij\in O_i}[p_{i(j-1)}z_{i(j-1)}\pi_{ij} + \sum_{i'\in I\setminus\{i\}}\Big(\sum_{i'j'\in O_{i'}} p_{ij}z_{ij}y_{ij,i'j'}\Big)]$$

$$s.t. -\pi_{ij} + \pi_{i(j-1)} - \sum_{i'\in I}\Big(\sum_{i'j'\in O_{i'}} y_{ij,i'j'} - x_{i'j',ij}y_{i'j',ij}\Big) \le 0 \qquad \forall ij, i(j-1) \in O_i\setminus\{in_i\}, i \in I \tag{14.1}$$

$$\pi_{i(n_i-1)} + \sum_{i'\in I}\Big(\sum_{i'j'\in O_{i'}} x_{i'j',in_i}y_{i'j',in_i}\Big) \le 1 \qquad \forall i \in I \tag{14.2}$$

$$z_{ij} = 1 \qquad \forall ij \in O_{i_{fix}}, i \in I \tag{14.3}$$

$$\sum_{i\in I}\sum_{ij\in O_{i_{var}}} z_{ij} \le K \tag{14.4}$$

$$\pi_{ij} \ge 0 \qquad \forall ij \in O_i, i \in I$$

$$y_{ij,i'j'} \ge 0 \qquad \forall ij \in O_i, i'j' \in O_{i'}, i, i' \in I$$

$$z_{ij} \in \{0,1\} \qquad \forall ij \in O_i, i \in I$$

besitzt eine quadratische Zielfunktion und eine total unimodulare Matrix der Nebenbedingungen, wie wir im vorherigen Abschnitt schon gesehen haben. Das dualisierte Modell der ersten Entscheidungsstufe erhielt immer eine ganzzahlige Lösung, weswegen wir dem Modell 11 die Ganzzahligkeitsbedingungen

$$y \in \mathbb{Z}^+, \pi \in \mathbb{Z}^+.$$

ergänzen dürfen.

Bedeutung der Variablen und Nebenbedingungen der Max-Szenario-Formulierung

Es sind nicht nur die Behandlungszeiten p von Modell 11 fest gegeben, sondern auch die Reihenfolge x bereits festgesetzt. Mithilfe der Patientenreihenfolgen x wird eine vollständige Auswahl eines disjunktiven Graphen beschrieben. Die Patientenreihenfolgen sind als Pfade gegeben, jedoch beschreibt dies eine vollständige Auswahl eines disjunktiven Graphen, da anhand dieser Pfade alle übrigen disjunktiven Bogen gerichtet werden können. Diesen Graphen, der durch die Variable x und die so definierte vollständige Auswahl S definiert wird, bezeichnen wir mit $G(S)$.

Die binäre Variable z gibt an, ob eine Behandlung eintreten wird oder nicht. Nebenbedingung (14.4) fordert, dass maximal K zusätzliche Behandlungen eintreten werden. Die als Dualvariablen ins Modell gekommene Variablen π und y sind die weiteren Variablen dieser Formulierung. Da die Variablen π und y aus dem dualen Modell 10 entstammen und diese Matrix dieser Nebenbedingungen total unimodular war, können π und y auch ganzzahlig gefordert werden.

Die Variable π gehört zu den Ungleichungen, dass jede Behandlung pro Patient nur nach der Vorgängerbehandlung für denselben Patienten starten darf und y gehört zu der Ungleichung, dass jede Behandlung $i'j'$ nur nach einer Behandlung ij derselben Behandlungsart starten darf, die ein Vorgänger von $i'j'$ im disjunktiven Graphen $G(S)$ ist. Für jeden konjunktiven Bogen von Behandlung $i(j-1)$ zu Behandlung ij des disjunktiven Graphen existiert nun also eine Variable π_{ij} und für jeden fixierten disjunktiven Bogen von Behandlung ij zu einer Behandlung $i'j'$ existiert eine Variable $y_{ij,i'j'}$. Die Nebenbedingungen (14.1) und (14.2) entsprechen Flusserhaltsbedingungen an jeder Behandlung ij, wobei an den Senken des disjunktiven Graphen nach Nebenbedingung (14.2) jeweils eine Einheit ankommen muss. Man kann die Variablen also so interpretieren, dass π_{ij} Fluss, der zu Knoten ij von dem Vorgängerknoten $i(j-1)$ auf dem Behandlungspfad fließt, beschreibt und die Variable $y_{ij,i'j'}$ Fluss, der zu einem Knoten $i'j'$ über den Vorgängerknoten ij der Behandlungsart fließt, beschreibt. Die quadratische Zielfunktion summiert die Kosten eines solchen Flusses über die Bogen der Behandlungspfade und die Kosten über die Reihenfolgebogen der Behandlungsarten. Kapazitäten auf den Bogen sind nicht gegeben.

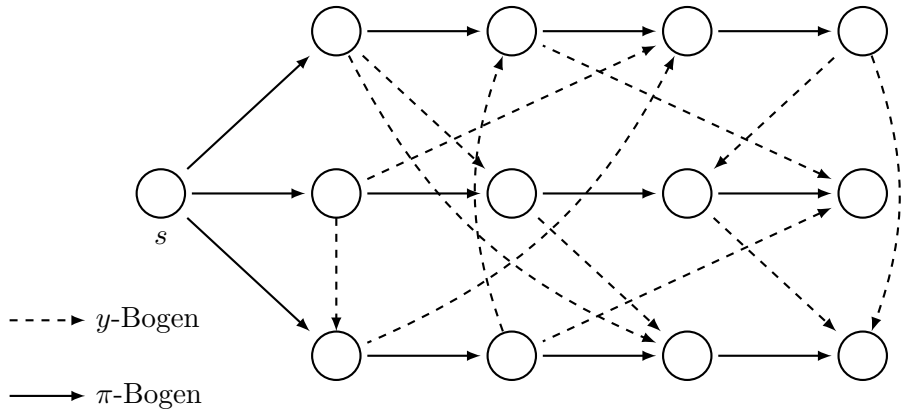


Abbildung 3.3: Die Dualvariablen π und y

In Abbildung 3.3 wird noch einmal verdeutlicht, welchen Bogen des disjunktiven Graphen den Dualvariablen π und y entsprechen. Die Variablen π stellen die konjunktiven Bogen dar und die Variablen y die gerichteten disjunktiven Bogen. Die ausgehenden Bogen an einem Knoten, der zu einer Behandlung O_{ij} gehört, haben Kosten von p_{ij}, sofern die Behandlung eintritt. Dies wird durch die quadratischen Terme der Zielfunktion $p_{ij}z_{ij}y_{ij,i'j'}$ für $ij \in O_i$ und $i'j' \in O_{i'}$ mit $i, i' \in I$ sowie $p_{i(j-1)}z_{i(j-1)}\pi_{ij}$ mit $ij \in O_i$, $i \in I$, deutlich. Falls die Behandlung nicht eintritt,

werden alle ausgehenden Bogen ohne Kosten berechnet. Außerdem müssen die Flussvariablen π und y mit den Kosten gewichtet werden.

Dieses Max-Szenario-Problem entspricht einem b-Fluss mit maximalen Kosten auf dem azyklischen Graphen $G(S)$, bei dem für K Behandlungen O_{ij}, deren Eintreten unsicher ist, die Behandlungsdauern von 0 auf p_{ij} erhöht werden. Dies ist eine Abwandlung des Flussproblems mit minimalen Kosten in azyklischen Digraphen. Die Balancen des Graphen $G(S)$ mit einer Quelle s und n Senken t_i, $i = 1, \ldots, n$ sind dabei $b(s) = n$, $b(t_i) = -1$ für alle $i = 1, \ldots, n$ und $b(v) = 0$ für alle übrigen Knoten.

Für die robuste Terminvergabe lässt sich das Problem nun wie folgt definieren.

Das Max-Szenario-Problem der robusten Terminvergabe unter unsicheren Behandlungspfaden

Instanz:	Eine vollständige Auswahl S eines disjunktiven Graphen $G = (V, C, F)$ unter unsicheren Behandlungspfaden mit Quelle s und n Senken $t_1, \ldots, t_n$ und einer Balancefunktion $b : V(G(S)) \to \mathbb{Z}$ mit $b(s) = n$, $b(t_i) = -1$ für alle $i \in I$ und $b(v) = 0$ sonst
Aufgabe:	Finde einen zulässigen b-Fluss mit maximalen Kosten, bei dem die Kosten von K Knoten $v_{ij} \in O_{i_{var}}$ von 0 auf p_{ij} erhöht werden dürfen

Ein solcher b-Fluss f mit K Knoten, deren Kosten erhöht wurden, kann durch das Tupel (f, B) beschrieben werden, wobei $B \subseteq V(G(S))$ die Menge der Knoten mit erhöhten Kosten ist.

Man kann dieses Max-Szenario-Problem allerdings auch etwas anders formulieren. Bisher wird ein kostenmaximaler b-Fluss gesucht, bei dem die Behandlungsdauer von K Behandlungen erhöht werden darf, wobei die Kosten auf den Knoten definiert sind. Wenn man nun einige Änderungen an dem azyklischen Graphen $G(S)$ vornimmt, lässt sich der Fluss auch über die Erhöhung von Bogenkosten definieren, so dass alle Behandlungsknoten fest sind.

(3.1) Definition (Modifizierter disjunktiver Graph)
Sei eine vollständige Auswahl S eines disjunktiven Graphen $G = (V, C, F)$ gegeben. Der modifizierte disjunktive Graph $G'(S) = (V', C' \cup S' \cup U)$, entsteht aus dem azyklischen Graphen $G(S)$ wie folgt:

- Erstelle für jeden Knoten $v \in V(G(S))$ einen Knoten v und für jeden Knoten $v \in V(G(S)) \backslash \{s\}$ eine zusätzliche Kopie v'
- Erstelle für alle Knoten $v \in V(G(S))$ einen Bogen vv' mit Kosten in Höhe der Behandlungsdauer der dazugehörigen Behandlung
- Erstelle für alle Bogen $vv' \in V(G(S))$, die zu einer nicht sicher eintretenden Behandlung gehören, einen weiteren Kostenwert von 0
- Erstelle für jeden Bogen $vw \in A(G(S))$ einen Bogen $v'w$ in $G'(S)$ mit Kosten von 0

- Erstelle für jeden Bogen $sv \in A(G(S))$ einen Bogen sv in $G'(S)$ mit Kosten von 0

Die Knotenmenge des Graphen $G'(S)$ bezeichnen wir mit V', die Menge der konjunktiven Bogen aus $G(S)$ in dem Graphen $G'(S)$ bezeichnen wir mit C', die Menge der gerichteten fixierten disjunktiven Bogen aus $G(S)$ mit S' und alle Bogen vv' mit $v \in V(G(S))$ liegen in der Bogenmenge U. Die Knoten $v \in V(G'(S))$ und $v' \in V(G'(S))$ erhalten jeweils die Balance $b(v)$ des Graphen $G(S)$. $\diamond$

Eine solche Reduktion einer vollständigen Auswahl eines disjunktiven Graphen ist in polynomieller Zeit durchführbar, da $2|V(G(S))|$ Knoten erstellt werden und $|A(G(S))| + |V(G(S))|$ Bogen. Indem man jedem Bogen mit nur einem Kostenwert denselben Kostenwert ein zweites Mal gibt, entsteht ein Digraph mit zwei Kosten für jeden Bogen. Die Kosten werden dabei für einen Bogen $a \in A(G')$ mit $(\underline{c}(a), \overline{c}(a))$ bezeichnet, wobei $\overline{c}(a) \geq \underline{c}(a)$ gilt. Ein Beispiel für einen modifizierten disjunktiven Graphen abgeleitet von der vollständigen Auswahl eines disjunktiven Graphen aus Abbildung 2.1, wird in der folgenden Abbildung 3.4 gezeigt.

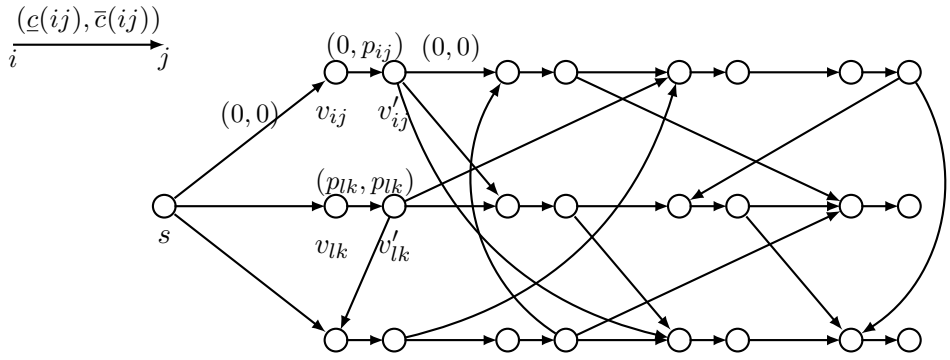


Abbildung 3.4: Beispiel eines modifizierten disjunktiven Graphen

Im Unterschied zu der Problemstellung in den gewöhnlichen disjunktiven Graphen sind hier die Kosten einer Behandlung, also die Behandlungsdauer, nicht mehr auf Knoten definiert, sondern auf dem Bogen, der zwischen zwei Knoten verläuft, die zu der gleichen Behandlung gehören.
In dem modifizierten disjunktiven Graphen $G'(S)$ mit Kosten $(\underline{c}(a), \overline{c}(a))$ für jeden Bogen $a \in A(G'(S))$ entspricht das Max-Szenario-Problem damit dem folgenden Problem.

Das Max-Szenario-Problem der robusten Terminvergabe unter unsicheren Behandlungspfaden in modifizierten disjunktiven Graphen

Instanz:	Ein modifizierter disjunktiver Graph $G'(S) = (V, A)$ unter unsicheren Behandlungspfaden mit einer Quelle $s \in V(G'(S))$ und Senken $t_i \in V(G'(S))$, $i \in I$ und einer Balancefunktion $b : V(G'(S)) \to \mathbb{Z}$ mit $b(s) = n$, $b(t_i) = -1$ für alle $i \in I$ und $b(v) = 0$ sonst
Aufgabe:	Finde ein Tupel (f, B), wobei f ein zulässiger b-Fluss mit maximalen Kosten ist und $B \subseteq A(G'(S))$ mit $\|B\| \leq K$ gilt, sodass für alle $a \in B$ $f(a) > 0$ ist und die Kosten $\overline{c}(a)$ gelten, und für alle übrigen Kanten $a' \neq B$ mit $f(a') > 0$ die Kosten $\underline{c}(a')$ verwendet werden

Dieses besondere Flussproblem mit maximalen Kosten, was in unserem Max-Szenario-Problem gesucht wird, lässt sich nicht nur für unsere Problemstellung definieren. Auch auf allgemeine azyklische Digraphen mit n Senken lässt sich unser Flussproblem übertragen, und zwar als Flussproblem mit maximalen Kosten, bei dem man bei K Bogen die Kosten erhöhen darf. Welche Bogen vom Kostenwert erhöht werden, gehört dabei zum Teil des Problems. Dieses werden wir für allgemeine azyklische Digraphen mit n Senken von nun an als das *Maximalkostenflussproblem mit K variablen Kosten* bezeichnen. Abkürzend bezeichnen wir dieses Problem auch als das K-MKF-Problem. Sei dazu $D = (V, A)$ ein azyklischer gerichteter Digraph mit einer Quelle s und n Senken $t_1, \ldots, t_n$. Für jeden Bogen $a \in D(A)$ sind zwei Kostenwerte $\underline{c}(a)$ und $\overline{c}(a)$ mit $\overline{c}(a) \geq \underline{c}(a)$ definiert. Das Maximalkostenflussproblem mit K variablen Kosten ist ein Flussproblem mit maximalen Kosten in D, wobei bei K Bogen a in D die Kosten von $\underline{c}(a)$ auf $\overline{c}(a)$ erhöht werden dürfen. Alle übrigen Bogen $a' \in A(D)$ müssen mit den niedrigeren Kosten $\underline{c}(a')$ berechnet werden.

Das Maximalkostenflussproblem mit K variablen Kosten

Instanz:	Ein azyklischer Digraph $D = (V, A)$, eine Quelle $s \in V(D)$, Senken $t_i \in V(D)$, $i \in I$, zwei Kostenfunktion $\overline{c} : A(D) \to \mathbb{Z}^+$ und $\underline{c} : A(D) \to \mathbb{Z}^+$ mit $\overline{c}(a) \geq \underline{c}(a)$ für alle $a \in A(D)$, eine Balancefunktion $b : V(D) \to \mathbb{Z}^+$ mit $b(s) = n$, $b(t_i) = -1$ für alle $i = 1, \ldots, n$ und $b(v) = 0$ sonst sowie $K \in \mathbb{Z}^+$
Aufgabe:	Finde ein Tupel (f, B), wobei f ein zulässiger b-Fluss mit maximalen Kosten ist und $B \subseteq A(D)$ mit $\|B\| \leq K$ gilt, sodass für alle $a \in B$ $f(a) > 0$ ist und die Kosten $\overline{c}(a)$ gelten, und für alle übrigen Kanten $a' \neq B$ mit $f(a') > 0$ die Kosten $\underline{c}(a')$ verwendet werden

In Kapitel 5 werden wir uns näher mit dem allgemeinen Maximalkostenflussproblem mit K variablen Kosten befassen.

Unser robuste Modell 11 für die Terminvergabe im Krankenhaus besitzt zwar lineare Nebenbedingung, jedoch ist die verwendete Zielfunktion

$$\max \sum_{i \in I} \Big(\sum_{ij \in O_i} [p_{i(j-1)} z_{i(j-1)} \pi_{ij} + \sum_{i' \in I \setminus \{i\}} \Big(\sum_{i'j' \in O_{i'}} p_{ij} z_{ij} y_{ij,i'j'} \Big)]$$

offensichtlich nicht linear, sondern quadratisch. Wir wollen hier in dieser Arbeit allerdings keine quadratischen Optimierungsprobleme betrachten. Wir können jedoch für das Maximalkostenflussproblem mit K variablen Kosten eine Modellierung betrachten, die linear und äquivalent zu Modell 11 ist.

Sei dafür $G = (V, C, F)$ ein disjunktiver Graph und S eine vollständige Auswahl S von G. Sei außerdem $G'(S) = (V', C' \cup S' \cup U)$ der entsprechende modifizierte disjunktive Graph. Die beiden Kosten, die auf den Bogen in der Menge U des modifizierten disjunktiven Graphen definiert sind, werden mit $\underline{c} : A(G'(S)) \to \mathbb{N}$ sowie $\overline{c} : A(G'(S)) \to \mathbb{N}$ notiert. Die Kosten der übrigen Bogen werden mit $\underline{c}$ beschrieben, wobei in unserem Fall der robusten Terminvergabe in Krankenhäusern $\underline{c}(a) = 0$ oder $\overline{c}(a) = p_{ij}$ für eine Behandlung O_{ij} gilt.

Modell 15 (Verwendete Modellierung des Max-Szenario-Problems)

$$\max \sum_{a \in C' \cup S' \cup U} \underline{c}(a) y(a) + \sum_{a \in U} (\overline{c} - \underline{c}(a)) w(a)$$

$$s.t. \sum_{a \in \delta^+(v)} y(a) - \sum_{a \in \delta^-(v)} y(a) = b(v) \qquad \forall v \in V(G'(S)) \tag{15.1}$$

$$\sum_{a \in U} z(a) \leq K \tag{15.2}$$

$$z(a) \leq y(a) \qquad \forall a \in U \tag{15.3}$$

$$w(a) \leq M z(a) \qquad \forall a \in U \tag{15.4}$$

$$w(a) \leq y(a) \qquad \forall a \in U \tag{15.5}$$

$$z(a) \in \{0, 1\} \qquad \forall a \in U \tag{15.6}$$

$$w(a) \in \mathbb{Z}^+ \qquad \forall a \in U \tag{15.7}$$

$$y(a) \in \mathbb{Z}^+ \qquad \forall a \in C' \cup S' \cup U \tag{15.8}$$

Die Variable $y(a)$ gibt an, wie viel Fluss über einen Bogen $a \in A(G'(S))$ mit Kosten $\underline{c}(a)$ fließt, und die binäre Variable $z(a)$ gibt an, ob die Kosten des Bogens $a \in U$ erhöht werden oder nicht. Zudem definiert die Variable $w(a)$, wie viel Fluss über einen Bogen $a \in U$ mit Kosten $\overline{c}(a)$ fließt. Die Konstante M ist ein großer konstanter Wert um zu gewährleisten, dass genug Fluss $w(a)$ über einen Bogen a fließen kann, beispielsweise $M = n$.

Diese Formulierung ist zwar linear in Zielfunktion und Nebenbedingungen, jedoch ist die Matrix A der Nebenbedingungen nicht total unimodular, wie nachfolgend zu sehen ist.

$$
A = \begin{array}{c}
\begin{array}{ccc} z & y & w \end{array} \\
\left(\begin{array}{c|c|c}
\overline{D} & & \\
\hline
 & 1 \dots 1 & \\
\hline
-\mathcal{I}_{|A(G'(S))|} & \mathcal{I}_{|A(G'(S))|} & \\
\hline
 & -M\mathcal{I}_{|A(G'(S))|} & \mathcal{I}_{|A(G'(S))|} \\
\hline
-\mathcal{I}_{|A(G'(S))|} & & \mathcal{I}_{|A(G'(S))|}
\end{array}\right)
\end{array}
$$

Hier beschreibe $\overline{D}$ die Inzidenzmatrix des modifizierten disjunktiven Graphen $G'(S)$ und $\mathcal{I}_{|A(G'(S))|}$ eine Einheitsmatrix der Größe $|A(G'(S))| \times |A(G'(S))|$. Durch die Nebenbedingung (15.4) ist diese Matrix nicht total unimodular. Das bedeutet, dass die Ganzzahligkeitsbedingungen der Variable z nicht relaxiert werden dürfen, ohne die Ganzzahligkeit einer optimalen Lösung zu verlieren.

Das Max-Szenario-Problem der robusten Terminvergabe im Krankenhaus unter unsicheren Behandlungspfaden entspricht nun dem Maximalkostenflussproblem mit K variablen Kosten in einem modifizierten disjunktiven Graphen. Wir zeigen im folgenden Kapitel 4, wie das Problem der robusten Terminvergabe unter unsicheren Behandlungspfaden insgesamt zu lösen ist, ohne explizit auf Lösungsverfahren für dieses Maximalkostenflussproblem mit K variablen Kosten einzugehen. In Kapitel 5 werden wir zu dem Maximalkostenflussproblem mit K variablen Kosten zurückkehren und es allgemein untersuchen, bevor wir anschließend in Kapitel 6 Lösungsverfahren dazu vorstellen werden.

4 Lösungsverfahren für die robuste Terminvergabe unter unsicheren Behandlungspfaden

Im vorherigen Kapitel haben wir auf der Basis unserer Problemstellung ein zweistufiges robustes Optimierungsmodell erstellt. Die beiden Entscheidungsstufen umfassten das Fixieren der disjunktiven Bogen eines disjunktiven Graphen und das Max-Szenario-Problem als Maximalkostenflussproblem, bei dem für maximal K Behandlungen, deren Eintreten unsicher sind, die Behandlungsdauern erhöht werden dürfen. Dieses Max-Szenario-Problem definierten wir als Maximalkostenflussproblem mit K variablen Kosten auf dem modifizierten disjunktiven Graphen.

Das Maximalkostenflussproblem mit K variablen Kosten, was wir im Folgenden meist nur noch als das K-MKF-Problem bezeichnen werden, werden wir in Kapitel 5 für allgemeine azyklische Digraphen ausführlich betrachten, um in Kapitel 6 zwei Verfahren kennenzulernen, mit denen sich das K-MKF-Problem in diesen Digraphen lösen lässt. Um unser Problem der robusten Terminvergabe unter unsicheren Behandlungspfaden aber allgemein zu lösen, wollen wir zunächst nicht explizit auf Lösungsverfahren für das K-MKF-Problem eingehen.

Unser robustes Modell 12

$$\min_{x} \max_{z,y,\pi} \sum_{i\in I}\Big(\sum_{ij\in O_i}\big[p_{i(j-1)}z_{i(j-1)}\pi_{ij} + \sum_{i'\in I\setminus\{i\}}\Big(\sum_{i'j'\in O_{i'}} p_{ij}z_{ij}y_{ij,i'j'}\Big)\big]$$

$$s.t. \sum_{i'\in I\setminus\{i\}}\sum_{\substack{i'j'\in O_{i'}\cup\{R\}\\ \mu_{ij}=\mu_{i'j'}}} x_{ij,i'j'} = 1 \qquad \forall ij\in O_i, j<n_i, i\in I$$

$$-\pi_{ij} + \pi_{i(j-1)} - \sum_{i'\in I}\Big(\sum_{i'j'\in O_{i'}} y_{ij,i'j'} - x_{i'j',ij}y_{i'j',ij}\Big) \le 0 \qquad \forall ij, i(j-1)\in O_i\setminus\{in_i\}, i\in I$$

$$\pi_{i(n_i-1)} + \sum_{i'\in I}\Big(\sum_{i'j'\in O_{i'}} x_{i'j',in_i}y_{i'j',in_i}\Big) \le 1 \qquad \forall i\in I$$

$$z_{ij} = 1 \qquad \forall ij\in O_{i_{fix}}, i\in I$$

$$\sum_{i\in I}\sum_{ij\in O_{ivar}} z_{ij} \le K$$

$$x_{ij,i'j'}\in\{0,1\} \qquad \forall ij\in O_i, i'j'\in O_{i'}, i,i'\in I$$

$$\pi_{ij}\ge 0 \qquad \forall ij\in O_i, i\in I$$

$$y_{ij,i'j'}\ge 0 \qquad \forall ij\in O_i, i'j'\in O_{i'}, i,i'\in I$$

$$z_{ij}\in\{0,1\} \qquad \forall ij\in O_i, i\in I$$

welches wir in Kapitel 3 bereits kennengelernt haben, beinhaltet die zwei Stufen der robusten Terminvergabe. Eine kompakte lineare Formulierung dieser robusten Modellierung haben wir

bislang nicht gefunden, weswegen wir das Modell nun auf einem anderen Weg lösen wollen, nämlich in einem ebenfalls zweistufigen Lösungsverfahren. Wir haben als Instanz der robusten Terminvergabe einen disjunktiven Graphen $G = (V, C, F)$ gegeben. Für diesen sind die Knoten und Bogen jeweils fest gegeben, nur die Dauer einer Behandlung ist eventuell unsicher. In einer ersten unveränderlichen Entscheidung sollen die disjunktiven Bogen fixiert werden, und zwar so, dass dabei die Lösung des Max-Szenario-Problems über alle vollständigen Auswahlen S minimiert wird.

In jeder vollständigen Auswahl S des disjunktiven Graphen G und dem dadurch entstehenden modifizierten disjunktiven Graphen $G'(S) = (V', C' \cup S' \cup U)$ lösen wir, wie ebenfalls in Kapitel 3 bereits gesehen, das Max-Szenario-Problem in der Formulierung

Modell 16

$$\max \sum_{a \in A(G'(S))} \underline{c}(a) y(a) + \sum_{a \in U} (\overline{c}(a) - \underline{c}(a)) w(a)$$

$$s.t. \sum_{a \in \delta^+(v)} y(a) - \sum_{a \in \delta^-(v)} y(a) = b(v) \qquad \forall v \in V(G'(S)) \qquad (16.1)$$

$$\sum_{a \in U} z(a) \leq K \qquad (16.2)$$

$$z(a) \leq y(a) \qquad \forall a \in U \qquad (16.3)$$

$$w(a) \leq M z(a) \qquad \forall a \in U \qquad (16.4)$$

$$w(a) \leq y(a) \qquad \forall a \in U \qquad (16.5)$$

$$z(a) \in \{0, 1\} \qquad \forall a \in U \qquad (16.6)$$

$$w(a) \in \mathbb{Z}^+ \qquad \forall a \in U \qquad (16.7)$$

$$y(a) \in \mathbb{Z}^+ \qquad \forall a \in A(G'(S)) \qquad (16.8)$$

Wir suchen in dem disjunktiven Graphen G die vollständige Auswahl S, für die der Wert von Modell 16 minimal ist. Sei dazu $\mathcal{G}$ die Klasse der modifizierten disjunktiven Graphen aller möglichen vollständigen Auswahlen in G. Das Modell 12 ist offensichtlich äquivalent zu der Formulierung von Modell 17, da über die Variable x eine vollständige Auswahl des disjunktiven Graphen G festgelegt wird. Die Reihenfolgen x legen in einem disjunktiven Graphen zunächst nur gerichtete Pfade der disjunktiven Bogen fest, wodurch noch nicht alle disjunktiven Bogen gerichtet sind. Diese Pfade legen jedoch auch die Richtungen der übrigen disjunktiven Bogen fest. Anhand eines Pfades $v_1 v_2 \dots v_l$, der ein Pfad der Länge $|Q| - 1$ für die Menge Q aller Patienten, die eine bestimmte Behandlungsart erhalten sollen, ist, lassen sich alle übrigen Bogen zwischen den Behandlungen der Patienten wie folgt fixieren: Für zwei Behandlungsknoten v_i und v_j mit $i < j$ fixiere den ungerichteten disjunktiven Bogen $\{v_i, v_j\}$ zu $v_i v_j$.

Modell 17

$$\min_{G'(S)=(V',C'\cup S'\cup U)\in\mathcal{G}} \max \sum_{a\in A(G'(S))} \underline{c}(a)y(a) + \sum_{a\in U} (\overline{c}(a) - \underline{c}(a))w(a)$$

$$s.t. \sum_{a\in\delta^+(v)} y(a) - \sum_{a\in\delta^-(v)} y(a) = b(v) \qquad \forall v \in V(G'(S)) \quad (17.1)$$

$$\sum_{a\in U} z(a) \leq K \quad (17.2)$$

$$z(a) \leq y(a) \qquad \forall a \in U \quad (17.3)$$

$$w(a) \leq Mz(a) \qquad \forall a \in U \quad (17.4)$$

$$w(a) \leq y(a) \qquad \forall a \in U \quad (17.5)$$

$$z(a) \in \{0,1\} \qquad \forall a \in U \quad (17.6)$$

$$w(a) \in \mathbb{Z}^+ \qquad \forall a \in U \quad (17.7)$$

$$y(a) \in \mathbb{Z}^+ \qquad \forall a \in A(G'(S)) \quad (17.8)$$

Für eine optimale Lösung von Modell 17 wird eine vollständige Auswahl benötigt, die eine minimale optimale Lösung des K-MKF-Problems liefert. Um diese optimale Lösung von Modell 17 und damit auch von Modell 12 zu finden, müssen alle möglichen vollständigen Auswahlen betrachtet werden und für diese das jeweilige Max-Szenario-Problem gelöst werden. Um alle dabei entstehenden modifizierten disjunktiven Graphen durchzugehen, bietet sich eine Art Branch-and-Bound-Algorithmus an. In Brucker [8] wird ein Branch-and-Bound-Algorithmus vorgestellt, der eine Lösung des Job Shop Scheduling Problems unter der Zielfunktion, dass der Makespan minimiert wird, definiert. Dieser ist für unser Unterproblem, dem Lösen des K-MKF-Problems, nicht sonderlich geeignet, weswegen wir in diesem Abschnitt nun einen anderen Lösungsalgorithmus für unser Problem vorstellen wollen.

In Abbildung 4.1 sehen wir dazu noch einmal den disjunktiven Graphen aus Abbildung 2.1.

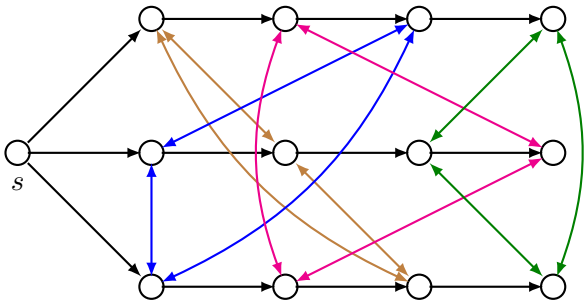


Abbildung 4.1: Disjunktiver Graph

Bei der Suche nach einer vollständigen Auswahl eines disjunktiven Graphen müssen die disjunktiven Bogen fixiert werden. Das bedeutet insbesondere, dass jeder Graph $G \in \mathcal{G}$ die folgende

Grundstruktur aus Abbildung 4.2 besitzt, die die konjunktiven Bogen bestimmen.

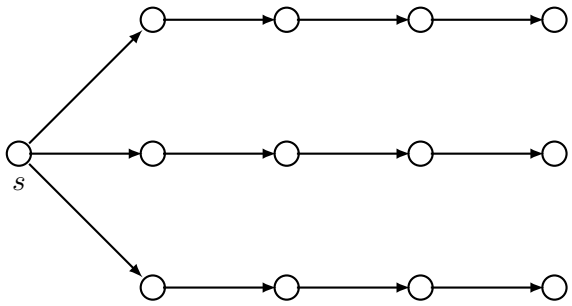


Abbildung 4.2: Grundstruktur der konjunktiven Bogen eines disjunktiven Graphen

Alle Behandlungen, die derselben Behandlungsart angehören, sind im disjunktiven Graphen über die disjunktiven Bogen zunächst als ungerichtete Cliquen benachbart, farblich dargestellt in Abbildung 4.1. Es gilt nun die Cliquenbogen so zu richten, dass eine vollständige Auswahl dabei entsteht. Eine vollständige Auswahl eines disjunktiven Graphen hat die Eigenschaften, dass jeder disjunktive Bogen gerichtet wurde und der dabei entstehende Graph azyklisch ist. Eine solche vollständige Auswahl wird in Abbildung 4.3 noch einmal dargestellt.

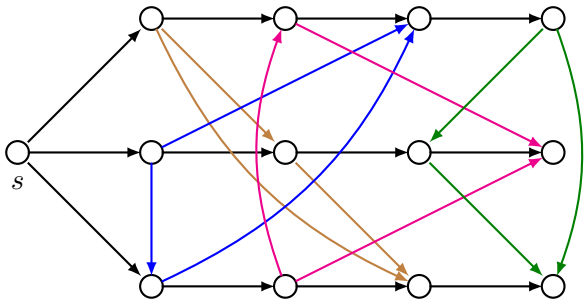


Abbildung 4.3: Beispiel einer vollständige Auswahl eines disjunktiven Graphen

Ganz allgemein sieht das hier vorgestellte Lösungsverfahren der Terminplanung im Krankenhaus unter unsicheren Behandlungspfaden wie folgt aus:

Algorithmus 2 Lösungsverfahren der robusten Terminvergabe unter unsicheren Behandlungspfaden

Input: ein disjunktiver Graph $G = (V, C, F)$ einer Instanz der robusten Terminvergabe unter unsicheren Behandlungspfaden

Output: eine robuster Terminplan unter unsicheren Behandlungspfaden

```
begin
    for jede vollständige Auswahl S des disjunktiven Graphen G do
        Erstelle den modifizierten disjunktiven Graphen G'(S);
        Löse für G'(S) das K-MKF-Problem;
        return eine optimale Lösung des K-MKF-Problems für eine optimale vollständige Auswahl von G;
    end
end
```

Wie schon erwähnt, wollen wir Lösungsverfahren für das K-MKF-Problem erst in den Kapiteln 5 und 6 genauer betrachten. In diesem Kapitel wollen wir ein Vorgehen angeben, um alle möglichen vollständigen Auswahlen zu betrachten und durch Abschneiden von vollständigen Auswahlen die Laufzeit für das Verfahren von Algorithmus 2 zu verbessern.

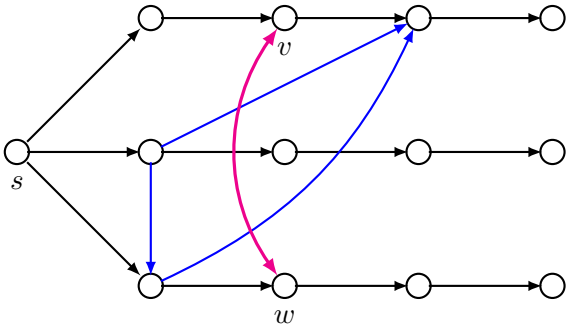
4.1 Branchingverfahren

Für jede vollständige Auswahl des durch die Problemstellung gegebenen disjunktiven Graphen müssen alle disjunktiven Bogen so fixiert werden, dass der dabei entstehende Graph azyklisch ist. Um alle möglichen vollständigen Auswahlen eines gegebenen disjunktiven Graphen zu bilden, speichern wir das Fixieren der disjunktiven Bogen in einem Suchbaum ab. In jedem Knoten des Suchbaums ist eine Auswahl S von disjunktiven Bogen fixiert worden. Der Suchbaum besitzt einen Wurzelknoten, in dem der initiale disjunktive Graph gespeichert wird, also gilt hier $S = \emptyset$. In den Nachfolgeknoten werden disjunktive Bogen fixiert, und dies rekursiv auf jeden Knoten des Suchbaums angewendet, bis eine vollständige Auswahl des zugrunde liegenden disjunktiven Graphen getroffen wurde. In den Blättern des Suchbaums sind die gesuchten vollständigen Auswahlen zu finden. Dies bedeutet insbesondere, dass nur auf den Graphen, die durch die vollständigen Auswahlen der Blätter des Suchbaums definiert werden, das K-MKF-Problem gelöst werden muss.

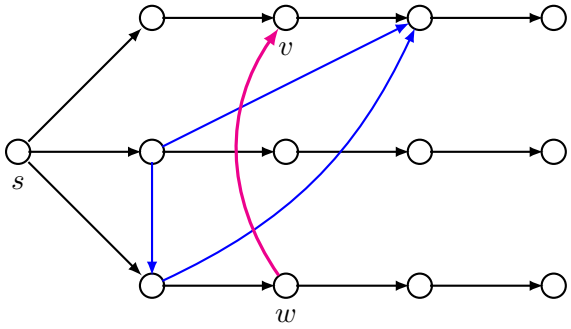
Formal definiert sich der Suchbaum nach Brucker [8] wie folgt: Ein Knoten r in einem Suchbaum definiert einen Graphen $G(S_r) = (V, C \cup S_r)$, wobei S_r die Menge aller fixierten disjunktiven Bogen in Knoten r symbolisiert. $Y(r)$ beschreibt alle möglichen Lösungen, die aus dem Graphen des Knotens r erzeugt werden können. Der Suchbaum wird nun an Knoten r verzweigt, in dem man die Menge aller möglichen Lösungen $Y(r)$ in q disjunkte Teilmengen $Y(s_1), \ldots, Y(s_q)$

zerteilt. Dabei entspricht wieder jedes $Y(s_i)$ einer Lösungsmenge eines Graphen $G(F_{s_i}) = (V, C \cup S_{s_i})$ mit $S_r \subset S_{s_i}$. Den so erstellten Suchbaum bezeichnen wir mit T.

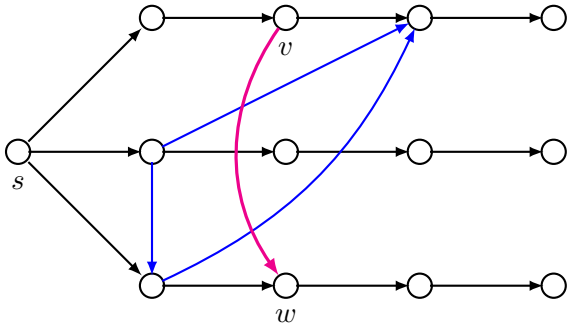
Von dem Suchbaum T benötigen wir für die Berechnung des Max-Szenario-Problems nur die Blätter. Alle übrigen Knoten werden nur benötigt, um alle möglichen vollständigen Auswahlen zu erzeugen und zu gewährleisten, dass keine vollständige Auswahl mehrfach betrachtet wird. Daher reicht es aus, in dem Suchbaum nur die Menge der fixierten disjunktiven Bogen zu speichern. Sei F die Menge aller disjunktiven Bogen des disjunktiven Graphen G und sei $\overline{F}$ eine beliebig geordnete Menge dieser disjunktiven Bogen. In dieser Reihenfolge sollen die disjunktiven Bogen nun nach und nach fixiert werden. Nach obiger Notation gilt also $q = 2$. Das bedeutet, dass in jedem Knoten des Suchbaums r, in dem schon i disjunktive Bogen fixiert wurden, zwei Nachfolgeknoten s_1 und s_2 erzeugt werden, in denen zusätzlich der $(i+1)$-te Bogen von $\overline{F}$ gerichtet wird. Für das Richten eines Bogens gibt es zwei Möglichkeiten, wobei jeweils eine dieser Möglichkeiten in einem der Nachfolgeknoten gespeichert wird.



(a) Ursprungsgraph D_r



(b) Verzweigung 1: D_{s_1}



(c) Verzweigung 2: D_{s_2}

Abbildung 4.4: Verzweigung im Suchbaum T zu einem disjunktiven Graphen

In Abbildung 4.4 wird ausgehend von einem Graphen D_r in Teilabbildung 4.4a, in dem bereits disjunktive Bogen fixiert wurden, gezeigt, wie eine Verzweigung aussieht. Hier wird der disjunktive Bogen $\{v, w\}$ gerichtet und somit entstehen die beiden Verzweigungen der Abbildungen 4.4b und 4.4c.

Für jeden Knoten r des Suchbaums T definieren wir eine untere Schranke $LB(r)$ und für das gesamte Branchingverfahren eine obere Schranke UB. Zu Beginn ist $UB = \infty$ und $LB(r) = -\infty$ für alle Suchbaumknoten $r \in V(T)$. Da wir insgesamt minimieren, gilt, dass jede zulässige Lösung eine obere Schranke ist. Untere Schranken geben einen Wert an, den eine Lösung des K-MKF-Problems in dem Graphen, den dieser Suchbaumknoten definiert, mindestens annimmt. Da die Lösung des K-MKF-Problems ein $b-$Fluss mit weiteren Eigenschaften ist, ist zum Beispiel die Lösung des Maximalkostenflussproblems, bei dem nur die unteren Kosten verwendet werden, eine untere Schranke für einen Suchbaumknoten.

Erst in den Blättern des Suchbaums wird das K-MKF-Problem gelöst. Nachdem dies für einen Blatt einmal berechnet wurde, geht man zu einem weiteren, noch nicht betrachteten Blatt b mit $LB(b) < UB$ des Suchbaums weiter und löst dort das entsprechende K-MKF-Problem. Falls der Graph einer vollständigen Auswahl in einem Blatt S_i des Suchbaums einen Kreis enthält, so ist $LB(S_i) = \infty$. Dies kann man etwa durch einen Suchbaum verhindern, der solche Graphen nicht erstellt. Ebenso könnte man den Suchbaum darauf beschränken, nicht alle disjunktiven Bogen zu fixieren, sondern nur die Knoten der Behandlungsarten zu einem Pfad zu richten.
In jedem Knoten r des Suchbaums wird eine untere Schranke $LB(r)$ angegeben, die einen Wert angibt, der kleiner oder gleich dem kleinstmöglichen Wert ist, der in der Lösungsmenge dieses Knotens noch möglich ist. Die Lösung eines Maximalkostenflussproblems mit K variablen Kosten in einem einzelnen Blatt des Suchbaums ist eine mögliche obere Schranke, da wir über alle vollständigen Auswahlen des disjunktiven Graphen minimieren wollen. Das bedeutet, dass eine neue Lösung des K-MKF-Problems in einem Blatt b die obere Schranke UB des Gesamtmodells möglicherweise verbessern kann, falls eine berechnete Lösung geringere Kosten als die aktuelle obere Schranke besitzt.

4.2 Abschneiden von Suchbaumknoten

Damit wir nicht jede einzelne vollständige Auswahl eines disjunktiven Graphen betrachten müssen, wollen wir in diesem Abschnitt einige Regeln betrachten, um die unteren Schranken einiger Suchbaumknoten zu verbessern. Dadurch können eventuell diese Suchknoten abgeschnitten werden und müssen nicht mehr betrachtet werden. Wir müssen für jede vollständige Auswahl das K-MKF-Problem lösen. Insgesamt suchen wir die vollständige Auswahl, die den Wert der Lösung des K-MKF-Problems minimiert. Unter bestimmten Umständen ist jedoch schon vor dem dazugehörigen Lösungsverfahren absehbar, dass eine zu betrachtende vollständige Auswahl keine Verbesserung der aktuellen oberen Schranke bringen kann, also dass für ein solches Suchbaumblatt r gilt, dass $LB(r) > UB$ ist.

Eine optimale Lösung des K-MKF-Problems ist ein Tupel (f, B), wobei f ein zulässiger b-Fluss in D ist und $B \subseteq A(D)$ eine Bogenmenge, für die gilt, dass $|B| \leq K$ ist, $f(a) > 0$ für alle $a \in B$ und $c(f) := c(f)_{|B} + c(f)_{|W}$ maximal ist mit $c(f)_{|B} := \sum_{a \in B} \overline{c}(a)$ und $c(f)_{|W} := \sum_{a \in B} \underline{c}(a)$ für $W := \{a \in A(D) | f(a) > 0, a \notin B\}$. Dies definieren wir nun als einen (b, K)-Fluss (f, B).

(4.1) Definition ((b, K)-Fluss)
Sei ein modifizierter disjunktiver Digraph $D = (V, A)$ einer vollständigen Auswahl eines disjunktiven Graphen gegeben mit einer Quelle $s \in V(D)$, n Senken $t_1, \ldots, t_n \in V(D)$ und einer Balancefunktion $b : V(D) \to \mathbb{Z}^+$ auf den Knoten $V(D)$ mit $b(s) = n$, $b(t_i) = -1$ für alle $i = 1, \ldots, n$ und $b(v) = 0$ sonst. Auf den Bogen des Digraphen D seien zwei Kostenfunktionen $\overline{c} : A(D) \to \mathbb{Z}^+$ und $\underline{c} : A(D) \to \mathbb{Z}^+$ definiert, wobei $\overline{c}(a) \geq \underline{c}(a)$ für alle Bogen $a \in A(D)$ gilt. Sei außerdem $K \in \mathbb{N}$.
Ein *zulässiger (b, K)-Fluss* (f, B) besteht aus ist einem zulässigen b-Fluss f und einer Bogenmenge $B \subseteq A(D)$ mit $|B| \leq K$, $f(a) > 0$ für alle $a \in B$ und $c(f) := c(f)_{|B} + c(f)_{|W}$, wobei $c(f)_{|B} := \sum_{a \in B} \underline{c}(a) f(a)$ und $c(f)_{|W} := \sum_{a \in B} \underline{c}(a) f(a)$ für $W := \{a \in A(D) | f(a) > 0, a \notin B\}$ gilt. ◇

Ein optimaler (b, K)-Fluss ist ein zulässiger (b, K)-Fluss (f, B) mit maximalen Kosten $c(f)$. Mithilfe eines solchen Flusses lassen sich die folgenden Regeln zur Schrankenverbesserung beweisen.

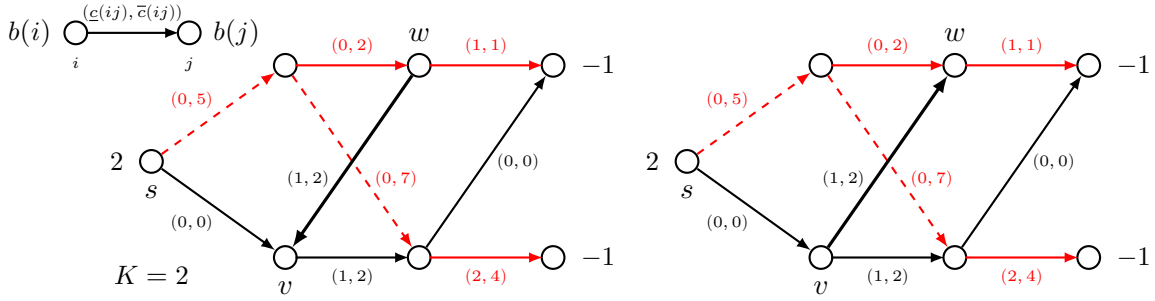
(4.2) Satz (Regel 1)
Sei $G'(S) = (V', C' \cup S' \cup U)$ ein modifizierter disjunktiver Graph, der auf einer vollständigen Auswahl S eines disjunktiven Graphen $G = (V, C, F)$ basiert und sei (f, B) ein optimaler (b, K)-Fluss einer Lösung des K-MKF-Problems für D . Zudem sei $\overline{S}$ mit $\overline{S} \subset S'$ die Menge an gerichteten disjunktiven Bogen, über die Fluss von f fließt, also $\overline{S} := \{a \in S' | f(a) > 0\}$. Dann gilt für jede optimale Lösung des K-MKF-Problems (f', B') eines modifizierten disjunktiven Graphen $G'(\overline{S'})$ mit $\overline{S} \subset \overline{S'}$ und $\overline{S'} \neq S'$

$$c(f') \geq c(f).$$ ◇

Beweis
Sei (f, B) ein (b, K)-Fluss mit maximalen Kosten in dem Graphen D und $\overline{S} := \{a \in S' | f(a) > 0\}$. In jedem übrigen Graphen $D' = (V, C' \cup \overline{S'} \cup U)$ mit $\overline{S} \neq \overline{S'}$ und $\overline{S} \subset \overline{S'}$ ist der Fluss (f, B) ebenfalls ein zulässiger (b, K)-Fluss. Für das K-MKF-Problem als Maximierungsproblem in dem Graphen D' gilt, dass die optimale Lösung mindestens die Kosten von (f, B) annimmt.■

In Abbildung 4.5 wird noch einmal verdeutlicht, dass der optimale Fluss (f, B), wobei f durch die roten Bogen eingezeichnet ist, auch in den Graphen, die dieselben fixierten disjunktiven Bogen besitzen, über die der b-Fluss f fließt, ein zulässiger (b, K)-Fluss ist. Die beiden Kostenwerte werden für jeden Bogen a in der Form $(\underline{c}(a), \overline{c}(a))$ notiert. Es soll je eine Flusseinheit von der Quelle s zu den beiden Senken versandt werden und dabei dürfen $K = 2$ Bogen erhöht werden. Teilabbildung 4.5a stellt einen optimaler (b, K)-Fluss in einem Graphen einer vollständigen Auswahl eines disjunktiven Graphen dar. Die erhöhten Bogen des (b, K)-Flusses werden in der Abbildung 4.5 durch die rot gestrichelten Bogen dargestellt. In der Teilabbildung

4.5b wird dieser (b, K)-Fluss in einer anderen vollständige Auswahl gezeigt, die sich von Abbildung 4.5a nur in einem Bogen $\{v, w\}$, über den kein Fluss fließt, unterscheidet. Für diesen Graphen ist der gezeigte (b, K)-Fluss (f, B) zulässig. Da nur Blätter r des Suchbaums mit $LB(r) < UB$ betrachtet werden, ist es anhand des Branchingschemas klar, dass diese Knoten im weiteren Branch-and-Bound-Verlauf nicht weiter betrachtet werden. An jedem Blatt p, dass die Menge $\overline{S}$, definiert wie in Satz 4.2, eines bisher berechneten (b, K)-Fluss (f, B) enthält, gilt $LB(p) \geq c(f)$. Da aber (f, B) eine zulässige Lösung ist, gilt damit auch $UB \leq c(f)$ gilt und es werden alle solchen Blätter p nicht weitere betrachtet. In dem Spezialfall von Satz 4.2, dass ein optimaler (b, K)-Fluss f sogar gar keine disjunktiven gerichteten Bogen verwendet, terminiert sofort das Branch-and-Bound-Verfahren.



(a) Optimaler (b, K)-Fluss in D

(b) Zulässiger (b, K)-Fluss in D'

Abbildung 4.5: Verdeutlichung von Satz 4.2

(4.3) Folgerung (Regel 2)
Sei $D = (V, C' \cup S' \cup U)$ ein modifizierter disjunktiver Graph einer vollständigen Auswahl eines disjunktiven Graphen $G = (V, C, F)$ und sei (f, B) ein optimaler (b, K)-Fluss als Lösung des K-MKF-Problems. Falls f nur konjunktive Bogen verläuft, also $\{a \in S' | f(a) > 0\} = \emptyset$ ist, kann der Branch-and-Bound-Algorithmus abgebrochen werden und f ist die optimale Lösung von Modell 12. ◇

Beweis
Der hier beschriebene (b, K)-Fluss (f, B) ist in jeder vollständigen Auswahl des disjunktiven Graphen ein zulässiger (b, K)-Fluss, also gilt für alle vollständigen Auswahlen des disjunktiven Graphen mit optimaler Lösung (f^*, B^*), dass $c(f^*) > c(f)$ ist. Somit löst f das Modell 12 optimal. ■

Damit der Algorithmus möglichst schnell terminiert, wollen wir die unteren Schranken $LB(r)$ für jedes Blatt r des Suchbaums, was einer vollständigen Auswahl S_r entspricht, möglichst schnell erhöhen und gleichzeitig die globale obere Schranke UB möglichst schnell verringern.

Jede optimale Lösung des K-MKF-Problems für eine vollständige Auswahl eines disjunktiven Graphen stellt eine zulässige Lösung des Gesamtproblems 12 dar. Da für eine zulässige Lösung

f gilt, dass $UB \leq c(f)$ ist, möchte man also möglichst schnell Lösungen berechnen, deren Kosten möglichst gering sind. Mit Knotenauswahlregeln anhand der Graphenstruktur können eventuell frühzeitig gute Lösungen als obere Schranken gefunden werden.

Eine Möglichkeit, für einen Knoten r des Suchbaums eine untere Schranke $LB(r)$ zu berechnen, ohne dass der Knoten r eine eindeutige Lösung als vollständige Auswahl eines disjunktiven Graphen darstellt, ist es, in dem Graphen, den dieser Knoten bislang darstellt, also $G(S_r) = (V, C \cup S_r)$, einen kostenmaximalen gewöhnlichen Fluss zu berechnen. Dies ist mit polynomiellem Aufwand mithilfe des Algorithmus von Edmonds und Karp [10] möglich. Hierfür müssen jedoch alle Kosten negiert werden. Dies berechnet eine optimale Lösung des Maximalkostenflussproblems, da der zugrunde liegende Graph azyklisch ist und alle Kosten nun negativ. Wendet man dies jedoch für jeden Knoten des Suchbaums an, ist der Aufwand nicht mehr polynomiell, da es insgesamt $2^{|F|+1}$ Knoten im Suchbaum gibt, wobei F die Menge aller disjunktiven Bogen beschreibt. $|F|$ ist jedoch ein fester Wert, also ist der Aufwand insgesamt pseudopolynomiell. Man könnte darauf verzichten, diese Schranke nicht in jedem Knoten zu berechnen, sondern nur in ausgewählten Suchbaumknoten und die Schranken an alle nachkommenden Knoten weiterzureichen. Sinnvoll kann es zum Beispiel dann sein, wenn es große strukturelle Unterschied des aktuell zu betrachtenden modifizierten disjunktiven Graphen zu den bisher betrachteten Graphen gibt.

4.3 Verbessertes Lösungsverfahren

Mithilfe der vorherigen Ergänzungen ergibt sich nun das insgesamte Lösungsverfahren.

Algorithmus 3 Verbessertes Lösungsverfahren der robusten Terminvergabe unter unsicheren Behandlungspfaden

Input: ein disjunktiver Graph $G = (V, C, F)$ einer Instanz der robusten Terminvergabe unter unsicheren Behandlungspfaden
Output: ein robuster Terminplan unter unsicheren Behandlungspfaden;
begin
  Bilde den Suchbaum T zum disjunktiven Graphen G mit Blättern $S_1, \ldots, S_l$;
  Setze $UB = \infty$ und $LB(r) = -\infty$ für alle Knoten $r \in V(T)$;
  Berechne einen kostenmaximalen Fluss f in G und setze $LB(r) = c(f)$ für alle $r \in V(T)$;
  Erstelle den modifizierten disjunktiven Graphen $G'(S_1)$;
  Löse das K-MKF-Problem für $G'(S_1)$;
  Sei f die optimale Lösung;
  if $c(f) < UB$ **then**
    Setze $UB = c(F)$ und verbessere die unteren Schranken;
  end
  while *noch Blätter r des Suchbaums existieren mit $LB(r) < UB$* **do**
    Gehe zum nächsten Blatt r von Suchbaum T mit $LB(r) < UB$;
    Erstelle den modifizierten disjunktiven Graphen $G'(S_r)$;
    Löse das K-MKF-Problem für $G'(S_1)$;
    Sei f die optimale Lösung;
    if $c(f) < UB$ **then**
      Setze $UB = c(F)$ und verbessere die unteren Schranken;
    end
  end
  return *UB und eine optimale Lösung des K-MKF-Problems in einer optimalen vollständigen Auswahl;*
end

Nachdem wir bislang Lösungsverfahren des K-MKF-Problems mit Ausnahme des ganzzahligen linearen Programms vernachlässigt haben, widmen wir uns in den folgenden Kapiteln nun diesem Problem und weiteren Lösungsverfahren dazu.

5 Maximalkostenflussproblem mit K variablen Kosten

Nachdem wir in Kapitel 4 den gesamten Lösungsansatz der robusten Terminvergabe im Krankenhaus unter unsicheren Behandlungspfaden gesehen haben, ohne näher auf Lösungsverfahren für das Max-Szenario-Problem einzugehen, wollen wir dies in den folgenden zwei Kapiteln nachholen und das Maximalkostenflussproblem mit K variablen Kosten, abkürzend K-MKF-Problem, für allgemeine azyklisch Digraphen untersuchen.

Das Maximalkostenflussproblem mit K variablen Kosten

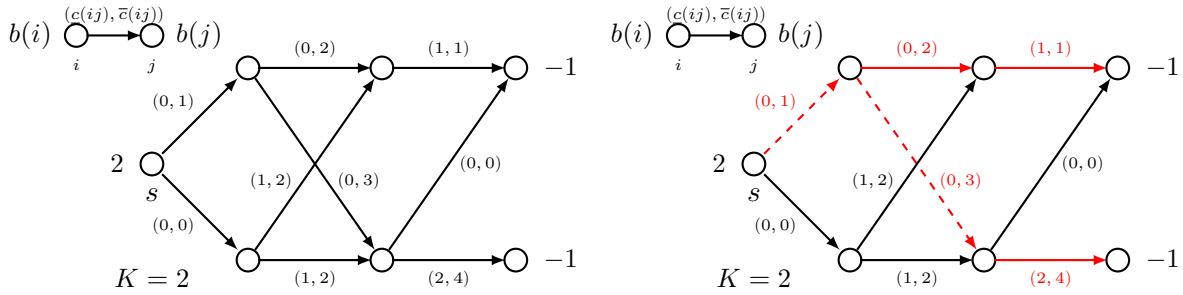
Instanz:	Ein azyklischer Digraph $D = (V, A)$ mit Quelle $s \in V(D)$, Senken $t_i \in V(D)$ mit $i = 1, \dots, n$, zwei Kostenfunktionen $\overline{c} : A(D) \to \mathbb{Z}^+$, $\underline{c} : A(D) \to \mathbb{Z}^+$ mit $\overline{c}(a) \geq \underline{c}(a)$ für alle $a \in A(D)$, einer Kapazitätsfunktion $u : A(D) \to \mathbb{Z}^+$, eine Balancefunktion $b : V(G) \to \mathbb{Z}^+$ mit $b(s) = n$, $b(t_i) = -1$ für alle $i = 1, \dots, n$ und $b(v) = 0$ sonst sowie $K \in \mathbb{Z}^+$
Aufgabe:	Finde ein Tupel (f, B) mit einem zulässigen b-Fluss f und einer Bogenmenge $B \subseteq A(D)$ mit $\lvert B\rvert \leq K$ mit $c(f) := c(f)_{\lvert B} + c(f)_{\lvert W}$ maximal, wobei $W := \{a \in A(D) \mid f(a) > 0, a \notin B\}$, $c(f)_{\lvert B} = \sum_{a \in B} \overline{c}(a) f(a)$ und $c(f)_{\lvert W} = \sum_{a \in W} \underline{c}(a) f(a)$ ist

Dieses Maximalkostenflussproblem mit K variablen Kosten ist nach besten Wissen des Autors in der Literatur bis dato noch nicht formuliert und näher untersucht worden. In diesem Kapitel werden wir dieses Problem nun aus theoretischer Sicht betrachten. Dazu werden wir noch einmal kurz auf das Problem eingehen und die Interpretation aus Kapitel 3 erläutern, einen Spezialfall betrachten und anschließend das Problem für allgemeine azyklische Digraphen untersuchen. Auch in diesem Kapitel bezeichnen wir das Maximalkostenflussproblem mit K variablen Kosten im Folgenden abkürzend als das K-MKF-Problem.

Das K-MKF-Problem entspricht der Suche nach einem b-Fluss in einem azyklischen Digraphen mit zwei Kostenfunktionen $\underline{c}$ und $\overline{c}$, für die gilt, dass $\overline{c}(a) \geq \underline{c}(a)$ für alle $a \in A(D)$ ist. Dieser Fluss f soll maximale Kosten besitzen und darf für maximal K Bogen, über die Fluss fließt, die Kosten von $\overline{c}$ verwenden und für alle übrigen Bogen die Kosten $\underline{c}$. Dieses entspricht dem in Kapitel 4.2 schon angesprochenen (b, K)-Fluss.

(5.1) Definition ((b, K)-Fluss)
Sei ein azyklischer Digraph $D = (V, A)$ gegeben mit Quelle $s \in V(D)$, Senken $t_1, \dots, t_n \in V(D)$ und einer Balancefunktion $b : V(D) \to \mathbb{Z}^+$ auf den Knoten mit $b(s) = n$, $b(t_i) = -1$ für alle $i = 1, \dots, n$ sowie $b(v) = 0$ für alle übrigen Knoten. Auf den Bogen des Digraphs D seien zwei Kostenfunktionen $\overline{c} : A(D) \to \mathbb{Z}^+$ und $\underline{c} : A(D) \to \mathbb{Z}^+$ mit $\overline{c}(a) \geq \underline{c}(a)$ für alle Bogen $a \in A(D)$ und eine Kapazitätsfunktion $u : A(D) \to \mathbb{Z}^+$ definiert. Sei außerdem $K \in \mathbb{N}$.
Ein *zulässiger (b, K)-Fluss* (f, B) besteht aus einem zulässigen b-Fluss f und einer Bogenmenge $B \subseteq A(D)$ mit $\lvert B\rvert \leq K$ und $f(a) > 0$ für alle $a \in B$. Es gilt für diesen (b, K)-Fluss,

dass die Kosten $c(f) = c(f)_{|B} + c(f)_{|W}$ entsprechen, mit $c(f)_{|B} := \sum_{a \in B} \overline{c}(a) f(a)$ und mit $c(f)_{|W} := \sum_{a \in B} \underline{c}(a) f(a)$ für $W := \{a \in A(D) | f(a) > 0, a \notin B\}$. ◇



(a) Digraph D

(b) Zulässiger (b, K)-Fluss in D

Abbildung 5.1: Beispiel eines (b, K)-Flusses

Ein solcher (b, K)-Fluss wird in Abbildung 5.1 gezeigt. Es soll jeweils eine Flusseinheit von der Senke s zu jeder Senke versendet werden. Der Fluss f wird durch alle rot gefärbten Bogen dargestellt und die gestrichelten roten Bogen stellen die Bogen da, deren Kosten mit $\overline{c}$ verwendet werden, also die Menge B.
Ein *maximaler* (b, K)*-Fluss* ist ein zulässiger (b, K)-Fluss, der unter allen möglichen (b, K)-Flüssen in einem solchen Digraphen D maximale Kosten besitzt. Offensichtlich entspricht nun das K-MKF-Problem der Suche nach einem optimalen (b, K)-Fluss in dem gegebenen azyklischen Digraphen D.

Das Problem, einen solchen optimalen (b, K)-Fluss zu finden, ist ein Maximierungsproblem. Jedoch lässt sich das Problem auch als Minimierungsproblem auffassen.

(5.2) Satz
Sei D ein azyklischer Digraph $D = (V, A)$ mit einer Senke $s \in V(D)$ und n Senken $t_1, \ldots, t_n \in V(D)$, mit zwei positiven Bogenkostenfunktionen $\overline{c} : A(D) \to \mathbb{Z}^+$, $\underline{c} : A(D) \to \mathbb{Z}^+$ mit $\overline{c}(a) \geq \underline{c}(a)$ für alle $a \in A(D)$ und einer Kapazitätsfunktion $u : A(D) \to \mathbb{Z}^+$ sowie $K \in \mathbb{N}$. Das Problem einen kostenmaximalen (b, K)-Fluss in diesem Graphen D zu finden ist äquivalent dazu, einen kostenminimalen (b, K)-Fluss in einem azyklischen Digraphen $\hat{D}$ zu finden mit $\hat{D} = (V(D), A(D))$, einer Kapazitätsfunktion $u : A(D) \to \mathbb{Z}^+$, zwei Bogenkostenfunktionen c_1 und $c_2 : A(\hat{D}) \to \mathbb{Z}^-$ mit $c_1 := -\overline{c}$, $c_2 := -\underline{c}$, $c_1(a) \leq c_2(a)$ für alle $a \in A(\hat{D})$. ◇

Beweis
Da der Graph D azyklisch und damit kreisfrei ist und da alle Kosten $\overline{c}(a)$ und $\underline{c}(a)$ größer gleich 0 sind, sind diese beiden Probleme äquivalent. ■

Man kann alle weiteren Ergebnisse auch über dieses Minimierungsproblem herleiten, wir verbleiben jedoch bei dem Maximierungsproblem. Im Kontext, den die Terminvergabe in Kran-

kenhäusern liefert, besitzt der gegebene azyklische Digraph D keine Kapazitätsfunktion, wie in Modell 12 zu erkennen ist. Im Vergleich zum allgemeinen Minimalkostenflussproblem existieren hier also keine Kapazitätsbedingungen.
Wir unterscheiden nun zwei Fälle. Zum einen azyklische Digraphen mit einer Senke und zum anderen azyklische Digraphen mit mehreren Senken.

5.1 Azyklische Digraphen mit einer Senke

Beginnen wollen wir unsere tiefergehenden Untersuchungen nun mit einem Spezialfall, nämlich dass es nur einen Knoten $v \in V(D)$ mit $b(v) < 0$ gibt. Der Graph besitzt also nur eine Senke. Wenn n die Anzahl an Senken angibt, gilt demnach $n = 1$. Für unsere betrachtete Krankenhausumgebung bedeutet dies, dass nur für einen Patienten ein robuster Zeitplan zu erstellen ist. Aber auch für diesen einen Patienten ist unklar, welche variablen Behandlungen auf seinem Behandlungspfad eintreten werden und welche nicht. Wir wissen nur, es sind maximal K zusätzliche Behandlungen, die zu den vorher bekannten Behandlungen eintreten können.

Für einen (b, K)-Fluss mit nur einer Senke soll genau eine Einheit von der Quelle s zu der einzigen Senke, bezeichnet mit $t \in V(D)$, versandt werden. Dies entspricht einem s-t-Weg in dem gegeben Digraphen D, wobei K der verwendeten Pfadbogen Kosten von $\overline{c}$ verwenden dürfen und alle übrigen Pfadbogen Kosten von $\underline{c}$.

(5.3) Satz
Für azyklische Digraphen D mit einer Quelle $s \in V(D)$, einer Senke $t \in V(D)$ und ganzzahligen Kapazitäten existiert immer ein ganzzahliger optimaler (b, K)-Fluss. ◇

Beweis
Sei (f, B) ein optimaler (b, K)-Fluss. Es wird eine Flusseinheit von Quelle s zu der Senke t verschickt. Wenn nun (f, B) nicht ganzzahlig ist, existieren mindestens zwei Teilwege P_1 und P_2, die bei demselben Knoten v_i starten und bei demselben Knoten v_j enden und es gilt, dass $f(P_1)$ und $f(P_2)$ nicht ganzzahlig sind. Sei ohne Einschränkung $c(P_1) \geq c(P_2)$ unter dem (b, K)-Fluss (f, B). Wenn man nun $\lceil f(P_1) \rceil$ Einheiten über den Weg P_1 verschickt und $\lfloor f(P_2) \rfloor$ Einheiten über den Weg P_2, dann entsteht ein zulässiger (b, K)-Fluss (f', B') mit $c(f') > c(f)$, was ein Widerspruch dazu war, dass (f, B) optimal war. Also existiert immer ein optimaler ganzzahliger (b, K)-Fluss in azyklischen Digraphen mit einer Senke. ■

Aufgrund von Satz 5.3 können wir annehmen, dass alle Kapazitäten größer als 0 sind, womit wir die Kapazitätsbedingungen vernachlässigen können. Das K-MKF-Problem für den Fall mit nur einer Senke ist damit das Folgende.

Das K-MKF-Problem für azyklische Digraphen mit einer Senke

Instanz:	Ein azyklischer Digraph $D = (V, A)$, Quelle $s \in V(D)$ und Senke $t \in V(D)$ zwei Kostenfunktionen $\overline{c} : A(D) \to \mathbb{Z}^+$, $\underline{c} : A(D) \to \mathbb{Z}^+$ mit $\overline{c}(a) \geq \underline{c}(a)$ für alle $a \in A(D)$ sowie $K \in \mathbb{Z}^+$
Aufgabe:	Finde einen zulässigen s-t-Weg P und eine Bogenmenge $B \subseteq A(P) \subseteq A(D)$, sodass alle Bogen $a \in B$ in P mit den Kosten $\overline{c}$ berechnet werden, alle übrigen Bogen mit Kosten von $\underline{c}$ und die Kosten von insgesamt P maximal sind

In den nächsten drei Abschnitten werden wir verschieden Algorithmen zur Lösung des K-MKF-Problems für azyklische Digraphen mit nur einer Senke behandeln. Dass nicht nur ein Ansatz betrachtet wird, liegt daran, dass verschiedene Betrachtungsweisen der Problemstellung zugrunde liegen und alle in dieser Arbeit vorgestellt werden sollen. Diese Ansätze lassen sich eventuell auch für andere Problemstellungen anwenden. Die drei Algorithmen sind jeweils polynomiell beschränkt, weswegen das hier betrachtete Problem mit einer Senke polynomiell lösbar ist.

5.1.1 Dynamisches Programm zur Lösung des K-MKF-Problems

Der erste dieser polynomiellen Algorithmen, den wir betrachten wollen, ist ein dynamisches Programm. Dabei beschreibe $l(v, k)$ die Kosten des teuersten v-t-Weges von einem Knoten $v \in V(D)\backslash\{t\}$ zu der Senke t in dem gegebenen azyklischen Digraphen D, auf dem insgesamt die Kosten von k Bogen erhöht wurden. Sei außerdem

$$N_v = \{w \in V(D) : vw \in A(D)\}$$

die Menge aller Nachfolger eines Knotens $v \in V(D)$.

Für den folgenden Algorithmus 4 ist es wichtig zu wissen, dass es nach Schrijver [22] für die Knotenmenge jedes azyklischen Digraphen D eine *topologische Sortierung* $v_1, \ldots, v_{|V(D)|}$ gibt, sodass für alle Bogen $v_i v_j$ gilt, dass $i < j$ ist. Eine solche topologische Sortierung lässt sich mit einem Aufwand von $\mathcal{O}(|V(D)| + |A(D)|)$ finden.

Die beiden Kosten $\overline{c}(a)$ und $\underline{c}(a)$, die für einen Bogen $a \in A(D)$ existieren, schreiben wir als Tupel $(\underline{c}(a), \overline{c}(a))$ für jeden Bogen $a \in A(D)$ auf.

Der Wert $f(v, k)$ gibt an, welcher Knoten w das Maximum der Rekursion für $l(v, k)$ angenommen hat. Anhand dieser Werte $f(v, k)$ lässt sich rekursiv beginnend bei $f(t, r)$ mit $r = \arg\max_{k \leq K} l(t, k)$ bestimmen, welcher s-t-Weg verwendet wird.

(5.4) Satz
Das Dynamische Programm beschrieben durch Algorithmus 4 löst das K-MKF-Problem in azyklischen Digraphen mit einer Senke optimal in $\mathcal{O}(|V(D)|^3)$. ◇

Algorithmus 4 Dynamisches Programm zur Lösung des K-MKF-Problems mit $n = 1$

Input: Azyklischer Digraph D mit Quelle s, Senke t und mit Bogenkosten $(\underline{c}(a), \overline{c}(a))\ \forall a \in A(G)$, $K \in \mathbb{N}$

Output: Die maximalen Kosten eines s-t-Weges P, für den K Bogen mit Kosten $\overline{c}$ verwendet werden dürfen

begin
  for $r \leq K$ **do**
    $l(t, r) = 0$;
  end
  forall $v = v_{|V(D)|-1}, \ldots, v_1$ **do**
    for $r \leq K$ **do**
      $l(v, k) = \max\limits_{w \in N_v} \max\{l(w, k) + \underline{c}(vw), l(w, k-1) + \overline{c}(vw)\}$;
      $f(v, k) = \arg\max\limits_{w \in N_v} \max\{l(w, k) + \underline{c}(vw), l(w, k-1) + \overline{c}(vw)\}$;
    end
  end
  return $L = \max\limits_{k \leq K} l(t, k)$ *und* f;
end

Beweis

Zunächst zeigen wir die Korrektheit des Dynamischen Programms.

Das Dynamische Progamm 4 löst das K-MKF-Problem in azyklischen Digraphen mit einer Senke optimal:

Die Korrektheit zeigen wir per Induktion.

Induktionsanfang: Offensichtlich ist $l(t, r) = 0$ für alle $r = 0, \ldots, K$, da von der Senke t keine Bogen abgehen.

Induktionsannahme: Wir nehmen an, dass die Werte $l(t, k), \ldots, l(w, k)$ sowie die Werte $l(v, 0)$, $\ldots, l(v, k-1)$ anhand der Rekursionsformel optimal berechnet wurden.

Induktionsschritt: Zu zeigen ist nun, dass die Rekursionsformel auch $l(v, k)$ optimal berechnet. Es soll gelten, dass

$$l(v, k) = \max_{w \in N_v} \max\{l(w, k) + \underline{c}(vw), l(w, k-1) + \overline{c}(vw)\},$$

ist, wobei N_v die Menge aller Nachbarn $w \in V(D)$ ist mit $vw \in A(D)$. Die Werte $l(w, k-1)$ und $l(w, k)$ sind nach Induktionsannahme optimal berechnet worden. Es wird der kostenmaximale v-t-Weg gesucht, bei dem maximal K Bogen mit Kosten von $\overline{c}$ verwendet werden dürfen. Es gibt zwei Möglichkeiten die maximalen Kosten eines v-t-Weges über einen

Nachbarn w zu bestimmen, nämlich $l(w,k) + \underline{c}(vw)$ sowie $l(w,k-1) + \overline{c}(vw)$. Da für alle positiven Nachbarn von v bereits alle optimalen Werte berechnet wurden, bestimmt also $\max\{l(w,k) + \underline{c}(vw), l(w,k-1) + \overline{c}(vw)\}$ einen kostenmaximalen v-t-Weg, der über den Nachbarn w führt. Daher bestimmt $\max_{w \in N_v} \max\{l(w,k) + \underline{c}(vw), l(w,k-1) + \overline{c}(vw)\}$ den kostenmaximalen v-t-Weg mit k Bogen, die Kosten von $\overline{c}$ verwenden.

Das Dynamische Programm 4 hat eine Laufzeit von $\mathcal{O}(|V|^3)$:

Es werden insgesamt maximal $|V(D)|K$ mögliche Werte berechnet, wobei $|V(D)| \leq K$ gilt, denn der Graph D ist kreisfrei und ein s-t-Weg kann daher über maximal $|V(D)| - 1$ Knoten verlaufen. Bei jedem dieser Paare werden maximal $2|V(D)|$ Berechnungen durchgeführt. Damit ergibt sich insgesamt eine Laufzeit von $\mathcal{O}(|V(D)|^3)$. ■

Den s-t-Weg, der die maximalen Kosten annimmt, ist über f zu ermitteln. Die Kosten, die auf einem daraufliegenden Bogen verwendet werden, sind über

$$\arg\max\{l(w,k) + \underline{c}(vw), l(w,k-1) + \overline{c}(vw)\}$$

für einen Bogen vw auf dem optimalen s-t-Weg zu ermitteln.
Mit dem Beweis von Satz 5.4 ist also gezeigt, dass das K-MKF-Problem für azyklische Digraphen mit einer Senke polynomiell lösbar ist.

(5.5) Folgerung
Das K-MKF-Problem für azyklische Digraphen mit einer Senke ist polynomiell lösbar.

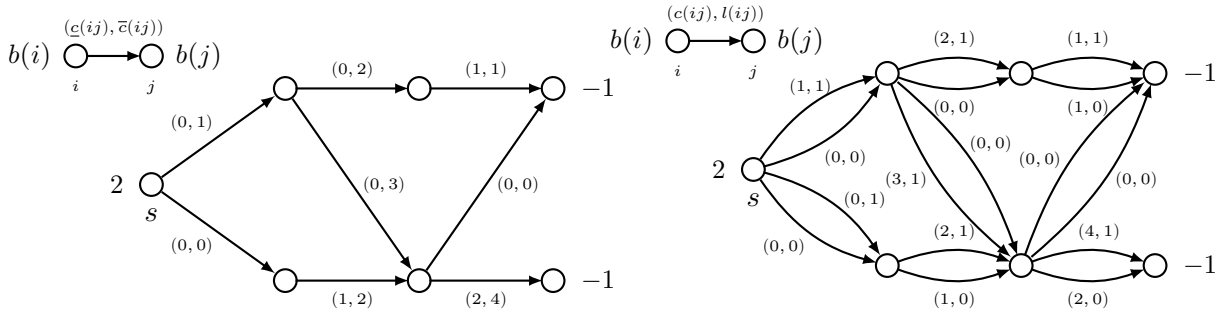
5.1.2 Längenbeschränkte kürzeste Wege zur Lösung des K-MKF-Problems

Eine weitere Möglichkeit, dass K-MKF-Problem in allgemeinen azyklischen Digraphen D mit nur einer Senke zu lösen, ist eine Variation des längenbeschränkten kürzesten Wege Problems. Die Idee dahinter ist, aus unserer Instanz des K-MKF-Problems einen Digraphen H zu erstellen, der jeden Bogen von D kopiert und eine Längenfunktion auf allen Bogen definiert. Durch die Bogenverdopplung erhält man einen Multidigraphen, der pro Bogen $a \in A(D)$ zwei Bogen a_1 und a_2 in $A(H)$ besitzt. Der Bogen a_1 erhält ein Gewicht von $\underline{c}(a)$ und eine Länge von 0 und der Bogen a_2 das Gewicht $\overline{c}(a)$ und Länge 1. Für den Gebrauch eines erhöhten Kostenwerts auf einem Bogen hat man also einen Längenverbrauch von einer Einheit. Die Längenschranke des so entstandenen Graphen entspricht dem Wert K.

(5.6) Definition (Längenbeschränkter Wege Graph)
Gegeben sei ein azyklischer Digraph $D = (V, A)$ mit einer Quelle s, einer Senke t sowie zwei Kostenfunktionen $\overline{c} : A(D) \to \mathbb{Z}^+$ und $\underline{c} : A(D) \to \mathbb{Z}^+$ mit $\overline{c}(a) \geq \underline{c}(a)$ für alle Bogen $a \in A(D)$.
Der *Längenbeschränkter Wege Graph* H entsteht aus dem Graphen D, indem man für jeden

Bogen $a \in A(D)$ einen Bogen $a_1 \in A(H)$ sowie eine Kopie $a_2 \in A(H)$ erzeugt. Der Bogen $a_1 \in A(H)$ erhält eine Länge von 0 und Kosten von $\underline{c}(a)$ und die Kopie a_2 erhält eine Länge von 1 und Kosten von $\overline{c}(a)$. ◇

Ein solcher Längenbeschränkter Wege Graph wird in der Abbildung 5.2 dargestellt. In D hat jeder Bogen $a \in A(D)$ die beiden Kostenwerte $\underline{c}(a)$ und $\overline{c}(a)$, die in Abbildung 5.2a als Tupel angegeben werden. In Abbildung 5.2b wird zu jedem dieser Kostenwerte ein Bogen identifiziert und eine Längenfunktion definiert. Aus einem schlichten Digraphen D ist nun ein Multidigraph H entstanden. Dabei besitzt der eigentliche azyklische Digraph für jeden Bogen $a \in A(D)$ zwei Kostenwerte $(\underline{c}(a), \overline{c}(a))$ und der Graph H als längenbeschränkter Wege Graph besitzt für jeden Bogen $a \in A(D)$ zwei Werte, nämlich Kosten $c(a)$ und eine Länge von $l(a)$.



(a) Digraph D (b) Längenbeschränkter Wege Graph H zu D

Abbildung 5.2: Der längenbeschränkte Wege Graph einer Instanz des K-MKF-Problems

Ziel ist es, einen s-t-Weg in H zu finden, der die Längenschranke einhält und maximale Kosten besitzt. Für die Lösung dieses Problems lässt sich der Labeling Dijkstra Algorithmus verwenden. Dieser findet eigentlich einen kürzesten, also kostenminimalen, längenbeschränkten Weg, weswegen das Problem zuvor angepasst werden muss. Dies kann darüber geschehen, die Bogenkosten des längenbeschränkten Wege Graphen H auf den negativen Wert zu ändern, also eine neue Kostenfunktion $c' : A(H) \to \mathbb{Z}$ zu definieren mit $c'(a) := -c(a)$.

(5.7) Satz
Der Labeling Dijkstra Algorithmus bleibt in azyklischen Digraphen korrekt, wenn alle Bogenkosten negativ sind. ◇

Beweis
Die Korrektheit des Labeling Dijkstras für positive Bogenkosten ist bewiesen. Da H ein azyklischer Digraph ist, existiert ein einfacher, optimaler längenbeschränkter kostenminimaler s-t-Weg in H. Insbesondere besitzt H keine Kreise und somit auch keine negativen.
Es sind alle Bogenkosten negativ. Sei $c_{\min} = \min_{a \in A(H)} c(a)$. Addiere auf die Kosten jedes Bogens den Wert $-c_{\min}$. Damit sind alle Bogenkosten größer gleich 0. In diesem Graphen H' berechnet nun der Labeling Dijkstra Algorithmus eine optimale Lösung P^*. Da alle Bogenkosten

mit derselben Konstante addiert wurde, ist der Weg P^* auch optimal für H. Die Kosten des entsprechenden Weges P berechnet über den Labeling Dijkstras mit negativen Bogenkosten entsprechen $c(P) = c(P^*) - c_{\min}|A(P^*)|$.

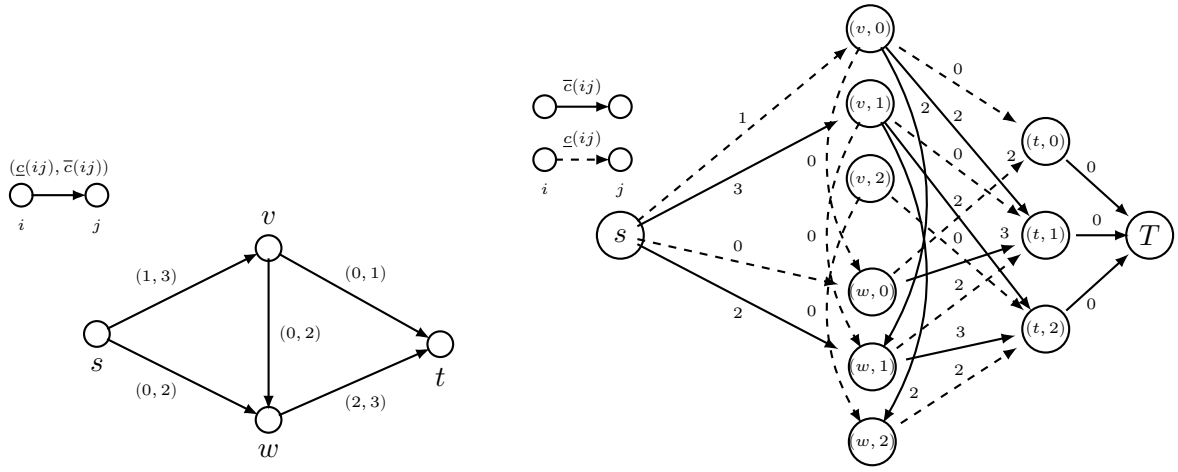
Der Labeling Dijkstra hat in diesem Fall, ähnlich wie das Dynamische Programm 4, eine Laufzeit von $\mathcal{O}(|V(H)|^3) = \mathcal{O}(|V(D)|^3)$, da $L = K \leq |V(D)|$ gilt.

5.1.3 Längste Wege zur Lösung des K-MKF-Problems

Wir stellen nun eine dritte Variante vor, die das K-MKF-Problem für azyklische Digraphen mit einer Senke t löst, nämlich über eine einfache Bestimmung des kostenmaximalen s-t-Weges in einem speziellen Graphen. Diesen Graphen, einen sogenannten *K-Längsten-Wege-Graphen*, erstellt man dazu aus dem gegebenen azyklischen Digraphen D wie folgt.

(5.8) Definition (*K*-Längsten-Wege-Graph)
Gegeben sei ein azyklischer Digraph $D = (V, A)$ mit einer Quelle $s \in V(D)$, einer Senke $t \in V(D)$ sowie zwei Kostenfunktionen $\overline{c} : A(D) \to \mathbb{Z}^+$ und $\underline{c} : A(D) \to \mathbb{Z}^+$ mit $\overline{c}(a) \geq \underline{c}(a)$ für alle Bogen $a \in A(D)$, sowie $K \in \mathbb{N}$.
Ein *K-Längster-Wege-Graph*, kurz KLW-Graph, H besitzt einen Quellknoten s, einen Senkeknoten T sowie $(|V(D)| - 1)(K + 1)$ weitere Knoten. Diese weiteren Knoten werden über die Knoten des Graphen D erstellt: für jeden Knoten $v \in V(D)\backslash\{s\}$ erstelle $K + 1$ Knoten in $V(H)$, bezeichnet mit (v, i), wobei $i = 0, \ldots, K$ gilt. Man verbindet nun für jeden Bogen $vw \in A(D)$ und für $r < K$ den Knoten (v, r) jeweils mit den Knoten (w, r) und $(w, r + 1)$. Der Bogen $(v, r)(w, r)$ erhält Kosten von $\underline{c}(vw)$ und der Bogen $(v, r)(w, r + 1)$ erhält die Kosten $\overline{c}(vw)$. Für $v = s$ entspricht (v, r) dem Knoten s. Für $r = K$ erstelle nur den Bogen $(v, r)(w, r)$ mit Kosten von $\underline{c}(vw)$. Alle Knoten (t, i) mit $i \leq K$ werden mit dem Senkeknoten T des Graphen H mit Kosten von 0 verbunden. ◇

Der neu erstellte KLW-Graph H ist wieder ein azyklischer Digraph mit einer Senke. Zur besseren Anschauung eines solchen K-längsten-Wege-Graphen hier nun ein explizites Beispiel eines solchen Graphen mit $K = 2$ in Abbildung 5.3. Zu dem Digraphen D aus Abbildung 5.3a ist der entsprechende KLW-Graph H in Abbildung 5.3b zu sehen, wobei $K = 2$ ist.



(a) Digraph D (b) 2-LW-Graph H

Abbildung 5.3: Beispiel eines 2-längsten-Wege-Graphen

In diesem neu erstellten Graphen H einer Instanz des K-MKF-Problems eines azyklischen Digraphen mit einer Senke findet man die Lösung des K-MKF-Problems über die Suche eines längsten, also kostenmaximalen, s-T-Weges.

Algorithmisch lässt sich dies im folgenden Algorithmus 5 beschreiben.

Algorithmus 5 Lösung des K-MKF-Problems über einen KLW-Graphen

Input: Azyklischer Digraph $D = (V, A)$ mit zwei Bogenkosten $(\underline{c}(a), \overline{c}(a))$ für alle $a \in A(D)$
Output: Lösung des K-MKF-Problems für den Graphen D
begin
  Erstelle den K-Längsten-Wege-Graphen H;
  Berechne einen längsten s-T-Weg P in H;
  return *P;*
end

(5.9) Satz
Der längste Weg in dem KLW-Graphen H entspricht der Lösung des K-MKF-Problems in einem azyklischen Digraphen mit einer Senke und ist in $\mathcal{O}(|V(D)|^3)$ berechenbar. ◇

Beweis
Es ist hier zu zeigen, dass der Algorithmus 5 eine optimale Lösung des K-MKF-Problems für den Graphen D berechnet und dafür insgesamt eine Laufzeit von $\mathcal{O}(|V(D)|^3)$ nötig ist.

Algorithmus 5 liefert eine optimale Lösung des K-MKF-Problems für azyklische Digraphen mit einer Senke:

Sei OPT_1 die optimale Lösung des K-MKF-Problems für einen solchen Digraphen D und OPT_2 die Länge des längsten Weges in dem entsprechenden KLW-Graphen H.

$OPT_2 \geq OPT_1$: Sei $P_1 = w_1 \dots w_l$ mit $w_1 = s$ und $w_l = t$ eine Lösung des K-MKF-Problems mit $k \leq K$ ausgewählten und erhöhten Bogen und $l(P_1) = OPT_1$. Dieser Weg ist ein längster s-t-Weg in D. Es existiert ein s-T-Weg P_2 im KLW-Graphen H, der die Länge $l(P_2) = OPT_1$ besitzt: Für alle Bogen $w_i w_{i+1}$, die als $r-ter$ Bogen auf dem Weg P_1 erhöht wurde , verläuft der Weg P_2 in H über den Bogen $(w_i, r)(w_{i+1}, r+1)$ und falls sie nicht erhöht wurde, aber auf dem Weg P_1 liegt, über den Bogen $(w_i, r)(w_{i+1}, r)$ mit $r \leq K$. Zusätzlich verläuft der Weg über den Bogen $(t, k)T$. Dies ist ein s-T-Weg in dem Graphen H mit einer Länge von OPT_1, also $OPT_2 \geq OPT_1$.

$OPT_2 \leq OPT_1$: Sei nun $Q_1 = s(u_1, k_1) \dots (u_l, k_l)T$ ein längster s-T-Weg in H mit einer Länge von $l(Q_1) = OPT_2$. Da maximal K Bogen erhöht wurden, existiert in D damit ein entsprechender Weg Q_2, der für einen Bogen $(u_i, k_1)(u_{i+1}, k_1 + 1) \in Q_1$ über den Bogen $u_i u_{i+1}$ verläuft und zwar mit Bogengewicht $\overline{c}(u_i u_{i+1})$ und mit einem Bogengewicht von $\underline{c}(u_i u_{i+1})$, falls der Weg über einen Bogen $(u_i, k_1)(u_{i+1}, k_1)$ verläuft. Dieser Weg Q_2 in D hat eine Länge von OPT_2 und maximal K erhöhten Bogen, womit $OPT_2 \leq OPT_1$ gilt.

Es gilt also insgesamt $OPT_1 = OPT_2$ und somit entspricht die Lösung des K-MKF-Problems genau der Lösung eines längsten Weges im entsprechenden KLW-Graphen H.

Algorithmus 5 besitzt eine Laufzeit von $\mathcal{O}(|V(D)|^3)$:

Der KLW-Graph H lässt sich mit einem Aufwand von $\mathcal{O}(|V(D)|^2)$ erstellen, denn es werden $K|V(D)| - K + 2$ Knoten erstellt und maximal $2K|A(D)|$ Bogen. Da ein s-t-Weg in D gesucht wird und D azyklisch ist, gilt $K \leq |V(D)|$. Also lässt sich der Graph H in einer Zeit von $\mathcal{O}(|V(D)|^3)$ erstellen. Die Berechnung eines längsten Weges in azyklischen Digraphen benötigt nach Schrijver [22] eine Laufzeit von $\mathcal{O}(|V(D)| + |A(D)|)$ und somit hat Algorithmus 5 eine insgesamte Laufzeit von $\mathcal{O}(|V(D)|^3)$. ■

In einem bereits erstellten KLW-Graphen H, dem eine Instanz des K-MKF-Problems zugrunde liegt, lässt sich mithilfe des folgenden Modells das Problem beschreiben.

Modell 18

$$\max \sum_{a \in A(H)} c(a)x(a) \tag{18.1}$$

$$s.t. \quad \sum_{a \in \delta^+(s)} x(a) = 1 \tag{18.2}$$

$$\sum_{a \in \delta^-(T)} x(a) = 1 \tag{18.3}$$

$$\sum_{a \in \delta^-(v)} x(a) - \sum_{a \in \delta^+(v)} x(a) = 0 \qquad \forall v \in V(H) \backslash \{s, T\} \tag{18.4}$$

$$x(a) \geq 0 \qquad \forall a \in A(H) \tag{18.5}$$

Die Bogenkosten bezeichnen wir der Einfachheit halber nur mit $c(a)$ für $a \in A(H)$. Da G azyklisch ist, bestimmt Modell 18 immer einen einfachen s-T-Weg. Die Nebenbedingungsmatrix, die dem Polyeder zugrunde liegt, ist total unimodular und somit nimmt eine optimale Lösung von Modell 18 immer einen ganzzahligen Wert an.

5.1.4 Das K-MKF-Problem im Krankenhauskontext mit einem Patienten

Wir wollen noch kurz darauf eingehen, was die in den vorherigen Abschnitten vorgestellten Lösungsverfahren für die robuste Terminvergabe in Krankenhäusern unter unsicheren Behandlungspfaden mit nur einer Senke bedeuten. Eine Senke bedeutet, dass nur für einen Patienten ein Terminplan erstellt werden muss. Allerdings ist auch für diesen Patienten unklar, welche der Behandlungen, deren Eintreten noch nicht sicher ist, wirklich auftreten werden. In unserem Max-Szenario-Fall sind die Kostenfunktionen $\overline{c}$ und $\underline{c}$ bei den Behandlungen gleich, bei denen das Eintreten sicher ist. Der modifizierte disjunktive Graph, der die folgende Gestalt wie in Abbildung 5.4 hat,

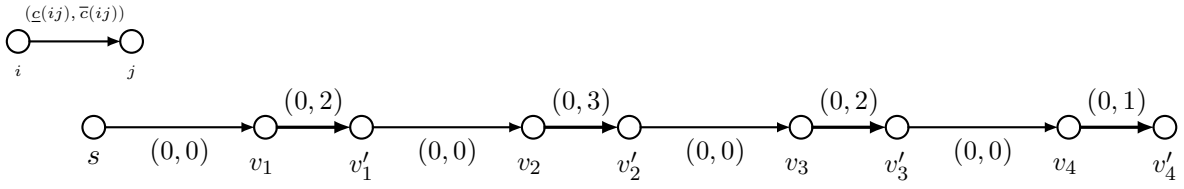


Abbildung 5.4: Beispiel eines modifizierten disjunktiven Graphen für $n = 1$

verdeutlicht, dass die Bogen, deren Kosten erhöht werden können, nur die zwischen den Knoten v und v' für alle $v \in V(D)$ seien können. Alle übrigen Bogen habe im Vorhinein feste Kosten, weswegen für diese keine Entscheidung über die Erhöhung der Kosten getroffen werden muss.

Da in der Beispielabbildung 5.4 alle Bogen $v_i v_i'$ für $i = 1, \ldots, 4$ unterschiedliche Kostenwerte für $\underline{c}$ und $\overline{c}$ besitzen, ist in diesem Beispiel also für alle Behandlungen unklar, ob sie eintreten werden oder nicht.

Für die Terminvergabe, die in dieser Arbeit untersucht wird, sind alle zuvor beschriebenen Lösungsverfahren aus den Abschnitten 5.1.1, 5.1.2 und 5.1.3 anwendbar, wir beschränken uns jedoch auf eine Lösung als längsten Weg im KLW-Graphen. Das Ziel dieses Abschnittes ist es, kurz eine Anpassung für die exakte Problemstellung zu erläutern. Dafür werden wir zunächst den KLW-Graphen, den wir hier verwenden wollen, für die genaue Problemstellung anpassen.

(5.10) Definition (Anpassung des KLW-Graphen)
Gegeben sei eine Instanz des Max-Szenario-Problems der robusten Terminvergabe unter unsicheren Behandlungspfaden für nur einen Patienten. Sei $D = (V, A)$ der modifizierte disjunktive Graph einer vollständigen Auswahl des eigentlichen disjunktiven Graphen mit Quelle s, Senke t sowie zwei Kostenfunktionen $\underline{c} : A(D) \to \mathbb{Z}^+$ und $\overline{c} : A(D) \to \mathbb{Z}^+$, wobei $\overline{c}(a) \geq \underline{c}(a)$ für alle Bogen $a \in A(D)$ gilt. Sei außerdem $K \in \mathbb{N}$.

Ein *K-Längster-Wege-Graph*, kurz KLW-Graph, H besitzt hier einen Quellknoten s, einen Senkeknoten T sowie $(|V(D)| - 1)(K + 1)$ weitere Knoten. Diese weiteren Knoten werden über die Knoten des Graphen D erstellt: für jeden Knoten $v \in V(D) \backslash \{s\}$ erstelle $K + 1$ Knoten in $V(H)$, bezeichnet mit (v, i) mit $i = 0, \ldots, K$. Man verbinde die Senke s mit allen Knoten $(w, 0)$, mit $vw \in A(D)$. Zudem erstelle man für alle (t, i), $i = 0, \ldots, K$ einen Bogen $(t, i)T$. Die bisher erstellten Bogen erhalten alle jeweils ein Gewicht von 0.

Für alle Bogen $vw \in A(D)$, wobei v und w für unterschiedliche Behandlungen stehen, erstellen wir die Bogen $(v, i)(w, i)$ für alle $i = 1, \ldots, K$ mit Kosten 0. Die verbleibenden Bogen in $A(D)$, die den Bogen vv' für alle Knoten $v \in V(D)$ entsprechen, werden nun wie folgt übertragen: Man verbindet jeden Knoten (v, r) mit den Knoten (v', r) und $(v', r + 1)$ für jeden Bogen $vv' \in A(D)$ mit $\overline{c}(vv') > \underline{c}(vv')$ und $r < K$. Die Bogen $(v, r)(v', r)$ erhält dabei Kosten von $\underline{c}(vv')$, der Bogen $(v, r)(v', r + 1)$ erhält Kosten $\overline{c}(vv')$. Für $r = K$ erstelle man den Bogen $(v, r)(v', r)$ mit Kosten von $\underline{c}(vv')$ und für alle $vv' \in A(D)$ mit $\overline{c}(vv') = \underline{c}(vv')$ $r < K$ erstelle man den Bogen $(v, r)(v', r)$ mit Kosten von $\underline{c}(vv')$. ◇

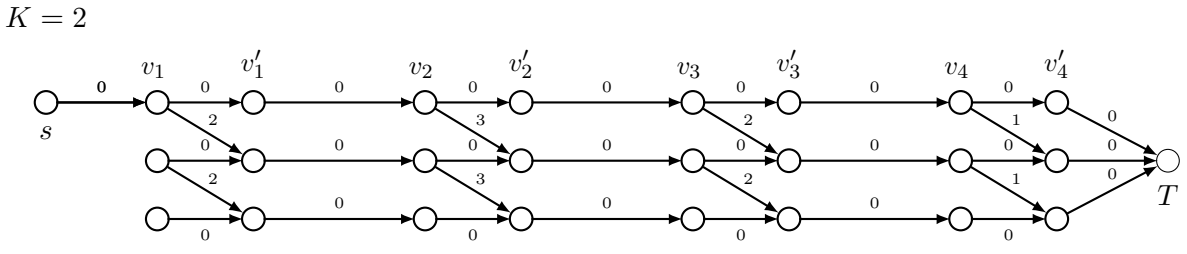


Abbildung 5.5: Beispiel eines 2-längsten Wege Graphen zu Abbildung 5.4

Analog zu Abschnitt 5.1.3 entspricht das K-MKF-Problem für einen Patienten im Kontext der robusten Terminvergabe unter unsicheren Behandlungspfaden in Krankenhäusern der Suche nach einem kostenmaximalen s-T-Pfad im KLW-Graphen H. In Abbildung 5.5 sehen wir einen solchen $2LW$-Graphen nach Definition 18 einer Probleminstanz ausgehend von Abbildung 5.4. Dieser Graph besitzt auch in der Anwendung für unser Krankenhausproblem maximal $|V(D)|K + 2$ Knoten und maximal $2|A(D)|K$ Bogen. Da, wie schon im vorherigen Abschnitt erläutert, $K \leq |V(D)|$ gilt, beträgt der Aufwand der Suche nach einem längsten Weg in diesem azyklischen Graphen maximal $\mathcal{O}(|V(D)|^3)$.

Die Annahme, dass nur ein Patient dem Krankenhaus vorliegt, entspricht offensichtlich nicht dem Normalfall, weswegen wir nun das K-MKF-Problem für allgemeine azyklische Digraphen mit n Senken betrachten wollen.

5.2 Azyklische Digraphen mit mehreren Senken

Wir gehen nun davon aus, dass es $n \geq 2$ Senken in dem zugrunde liegenden Digraphen gibt. Das robuste Modell 11 ist damit so zu interpretieren, dass im Unterschied zu dem Fall $n = 1$ nicht mehr ein längster Weg gesucht wird, sondern ein (b, K)-Fluss, der kostenmaximal ist. Einen kostenmaximalen Fluss in azyklischen Digraphen zu finden ist nach Edmonds und Karp [10] polynomiell lösbar, jedoch kann man aufgrund der Eigenschaft, dass K Bogen über diesen Fluss mit variablen Kosten gewählt werden können, damit die Kosten maximal sind, die dazu verwendeten Lösungsverfahren nicht einfach verallgemeinern. Aus diesem Grund werden wir hier in diesem Abschnitt (b, K)-Flüsse näher untersuchen.

(5.11) Definition (Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$)
Sei $D = (V, A)$ ein azyklischer Digraph mit einer Quelle $s \in V(D)$ und n Senken $t_1, \ldots, t_n \in V(D)$. Auf den Knoten $V(D)$ ist eine Balancefunktion $b : V(D) \to \mathbb{Z}$ definiert mit $b(s) = n$, $b(t_i) = -1$ für alle $i = 1, \ldots, n$ und $b(v) = 0$ für alle übrigen Knoten $v \in V(D)$. Auf den Bogen sind zwei Kostenfunktionen $\overline{c} : A(D) \to \mathbb{Z}^+$ und $\underline{c} : A(D) \to \mathbb{Z}^+$ mit $\overline{c}(a) \geq \underline{c}(a)$ für alle Bogen $a \in A(D)$ und eine Kapazitätsfunkion $u : A(D) \to \mathbb{Z}^+$ definiert. Zudem ist ein Wert $K \in \mathbb{N}$ gegeben, der angibt, wie viele Bogen mit Kosten $\overline{c}$ gewählt werden dürfen. Diese Instanz des K-MKF-Problems wird mit $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$ beschrieben.

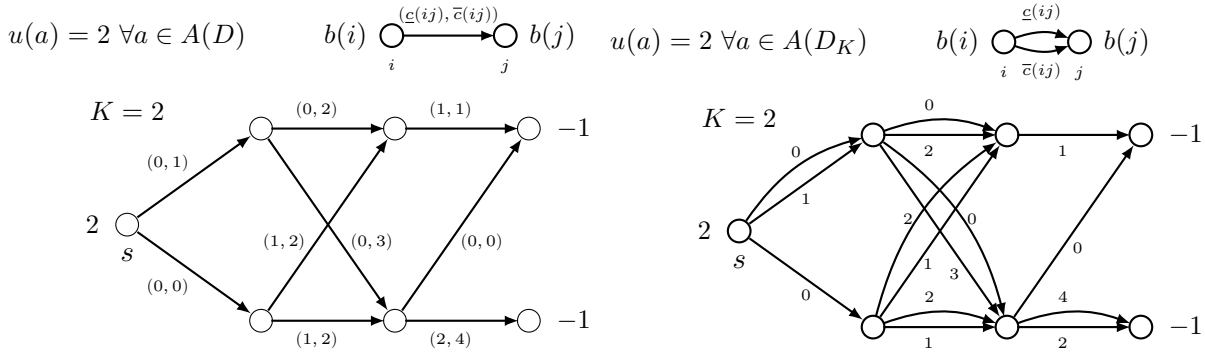
Eine entsprechende Instanz zum äquivalenten Minimierungsproblem aus Satz 5.2 mit negativen Kosten wird analog mit $(D, s, t_1, \ldots, t_n, -\overline{c}, -\underline{c}, u, b, K)$ bezeichnet.

Die Lösung eines K-MKF-Problems wird durch einen (b, K)-Fluss nach Definition 5.1 gegeben, wobei dieser (b, K)-Fluss maximale Kosten besitzt. Wir wollen nun, analog zu gewöhnlichen kostenminimalen Flüssen, ein Optimalitätskriterium angeben, das zeigt, ob ein zulässiger (b, K)-Fluss optimal ist. Dazu sei (f, B) ein zulässiger (b, K)-Fluss einer K-MKF-Probleminstanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$. Für die Herleitung des Optimalitätskriteriums

werden wir aus unserer Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$ und dem (b, K)-Fluss (f, B) eine weitere Graphendarstellung bauen.

(5.12) Definition (K-Graph)
Sei $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$ eine Instanz des K-MKF-Problems. Der *K-Graph* D_K entsteht aus dem azyklischen Digraphen D, indem wir für jeden Bogen $a \in A(D)$ mit $\overline{c}(a) > \underline{c}(a)$ einen Bogen $a_1 \in A(D_K)$ sowie eine Kopie $a_2 \in A(D_K)$ erzeugen. Die beiden Bogen a_1 und a_2 in $A(D_K)$ erhalten beide eine Kapazität von $u(a)$. Zudem erhält der Bogen $a_1 \in D_K$ Kosten von $\overline{c}(a)$ und der Bogen a_2 die Kosten $\underline{c}(a)$. ◇

Die Bogen $a \in A(D)$ mit $\overline{c}(a) = \underline{c}(a)$ bleiben unverändert und die Kostenfunktion des Graphen D_K bezeichnen wir im Folgenden mit c. Ebenso bleiben die Balancen der Knoten auch im K-Graphen bestehen. Einen solchen K-Graphen D_K sehen wir in der folgenden Abbildung 5.1 verdeutlicht. In dem eigentlichen Graphen D aus Abbildung 5.6a existieren für jeden Bogen $a \in A(D)$ zwei Kosten, die in dem Tupel $(\underline{c}(a), \overline{c}(a))$ für jeden Bogen $a \in A(D)$ zusammengefasst werden. Der K-Graph D_K, siehe Abbildung 5.6b, ist ein Multidigraph, bei dem jeder Bogen $a \in A(D_K)$ jedoch nur noch einen Kostenwert $c(a)$ besitzt, wobei für zwei parallele Bogen jede genau einen der Werte $\overline{c}$ und $\underline{c}$ erhält.



(a) Digraph D (b) K-Graph D_K

Abbildung 5.6: K-Graph D_K eines Digraphen D

Zur Lösung des K-MKF-Problems sucht man in dem K-Graphen D_K einen maximalen (b, K)-Fluss. Jedoch dürfen, da D_K ein Multidigraph ist, nicht alle Bogen zusammen verwendet werden. Für zwei parallele Bogen a und $a' \in A(D_K)$ darf ein (b, K)-Fluss (f, B) maximal über einen dieser Bogen Fluss schicken. Somit ergibt sich das K-MKF-Problem in einer geänderten Formulierung.

Das K-MKF-Problem in K-Graphen

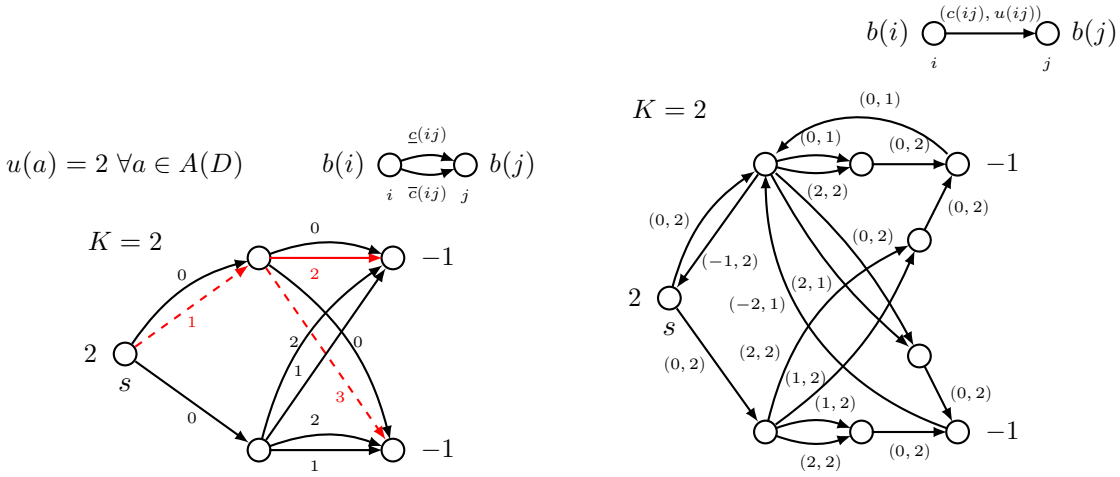
Instanz:	Der K-Graph D_K einer Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$ des K-MKF-Problems
Aufgabe:	Finde ein Tupel (f, B) in D_K mit einem zulässigen b-Fluss f und einer Bogenmenge $B \subseteq A(D)$ mit $\lvert B\rvert \leq K$, sodass $c(f) := c(f)_{\lvert B} + c(f)_{\lvert W}$ maximal ist mit $W := \{a \in A(D) \lvert f(a) > 0, a \notin B\}$, und $c(f)_{\lvert B} := \sum_{a \in B} \overline{c}(a) f(a)$ sowie $c(f)_{\lvert W} := \sum_{a \in W} \underline{c}(a) f(a)$. Von zwei parallelen Bogen a_1 und a_2 in $A(D_K)$ darf jedoch maximal einer in dem b-Fluss f enthalten sein

Ein solches Tupel (f, B) in dem K-Graphen D_K bezeichnen wir ebenfalls wieder als (b, K)-Fluss. Offensichtlich stimmt die Formulierung des K-MKF-Problems in K-Graphen mit der bisherigen Formulierung aber überein. Falls man nun einen zulässigen (b, K)-Fluss in dem Graphen D_K gegeben hat, lässt sich anhand eines Optimalitätskriteriums prüfen, ob der Fluss maximal ist. Vor der Angabe des Optimalitätskriterium 5.15 benötigen wir jedoch die Definition eines K-Residualgraphen.

(5.13) Definition (K-Residualgraph)
Sei (f, B) ein zulässiger (b, K)-Fluss einer Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$ des K-MKF-Problems. Den Residualgraphen $D_K(f)$ des K-MKF-Problems bildet man aus dem K-Graphen D_K der Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$ wie folgt: Für jeden Bogen $a \in A(D_K)$ mit $f(a) > 0$ setze die Kapazität von $a \in D_K(f)$ auf $u(a) - f(a)$. Für alle Bogen $a = vw \in A(D_K)$ mit $f(a) > 0$ und $f(a) < u(a)$, füge in $D_K(f)$ einen weiteren Knoten v_a ein. Füge einen Bogen wv_a mit Kapazität $u(a)$ und Kosten 0 ein und ändere die Bogen $wu \in A(D_K)$ mit $u \in D_K(f)$ zu Bogen $v_a u$ mit gleichbleibenden Kosten und Kapazitäten. Füge außerdem für alle Bogen $a = vw \in A(D_K)$ mit $f(a) > 0$ einen Residualbogen $v_a v$ ein mit Kosten von $-c(a)$ und einer Kapazität von $f(a)$ ein. ◇

Für eine bessere Anschauung wollen wir auch hier in Abbildung 5.7 ein Beispiel eines solchen K-Residualgraphen angeben.

In dieser Abbildung 5.7 sieht man zum einen in Abbildung 5.7a, wie ein (b, K)-Fluss (f, B) in dem K-Graphen D_K aussieht und zum anderen in Abbildung 5.7b, wie ein dazu entsprechender K-Residualgraph $D_K(f)$ aussieht. In dem K-Graphen D_K ist der b-Fluss f durch die rotgefärbten Bogen zu sehen. Die Menge B entspricht der rotgestrichelten Bogenmenge, diese Bogen werden also mit einem Bogengewicht von $\overline{c}$ bei der Kostenberechnung von f verwendet. Anhand des K-Residualgraphen $D_K(f)$ sieht man, dass es verschiedene Arten der Bogen gibt. Zum einen die beiden Bogen mit unterschiedlichen Bogenkosten, die auch der K-Graph enthält, aber nun zusätzlich noch Residualbogen. Da ein zulässiger (b, K)-Fluss maximal eine von zwei parallelen Bogen aus D_K verwenden darf, existiert in dem K-Residualgraphen $D_K(f)$ auch nur maximal ein Residualbogen. Jedoch kann dies ein Residualbogen zu einem Bogen mit Kosten von $\underline{c}$ oder mit Kosten von $\overline{c}$ sein.



(a) (b, K)-Fluss (f, B) in einem K-Graphen D_K (b) K-Residualgraph $D_K(f)$

Abbildung 5.7: K-Residualgraph $D_K(f)$ zu einem (b, K)-Fluss (f, B) in D_K

(5.14) Definition (K-Bogen)
Sei $D_K(f)$ ein K-Residualgraph bezüglich einer Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$ des K-MKF-Problems und eines dazugehörigen (b, K)-Flusses f.
Ein Bogen $a \in D_K(f)$ mit $c(a) = \overline{c}(a)$ wird als K- *Bogen* bezeichnet und ein Bogen $a \in D_K(f)$ mit $c(a) = -\overline{c}(a)$ wird als K-*Residualbogen* bezeichnet. Die Bogen $a \in D_K(f)$ mit $c(a) = \underline{c}(a)$ und $c(a) = -\underline{c}(a)$ werden als *neutrale Bogen* beziehungsweise als *neutrale Residualbogen* bezeichnet. Die Bogen $a \in D_K(f)$ mit $c(a) = 0$, die für je zwei parallele Bogen erzeugt werden, werden auch Hilfsbogen genannt. ◇

K-Bogen entsprechen also im K-Residualgraphen den Bogen mit den Kosten $\overline{c}$. Zudem sind K-Residualbogen und die neutralen Residualbogen die durch den Fluss entstandenen Residualbogen der entsprechenden Bogen.

Das Optimalitätskriterium um zu prüfen, ob ein (b, K)-Fluss (f, B) einer Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$ optimal ist, lässt sich nun wie folgt beschreiben.

(5.15) Satz (Optimalitätskriterium)
Sei $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$ eine Instanz des K-MKF-Problems und (f, B) ein zulässiger (b, K)-Fluss in D_K.
Der (b, k)-Fluss (f, B) ist genau dann optimal, wenn im entsprechenden K-Residualgraphen $D_K(f)$ keine Kreise $C_1, \ldots, C_r$ existieren, wobei

- $A_R(C_i)$ die Menge aller K-Residualbogen des Kreises C_i ist,
- $A_K(C_i)$ die Menge aller K-Bogen des Kreises C_i ist,

- $\alpha = |\bigcup_{i=1}^{r} A_K(C_i)|$,
- I_a der Indexmenge aller Kreise C_i entspricht, die über einen Bogen $a \in A(D_K(f))$ verlaufen,
- $\delta^i_{\min}$ der Vielfachheit eines Kreises C_i entspricht, so dass für jeden Bogen $a \in A(D_K(f))$ gilt, dass $\sum_{i \in I_a} \delta^i_{\min} \leq u(a)$ ist,

und für diese Kreise gilt, dass

$$|\bigcup_{i=1}^{r} A_K(C_i)| \leq |\bigcup_{i=1}^{r} A_R(C_i)|$$

ist und dass die Kosten

$$\sum_{i=1}^{r} c(C_i)\delta^i_{\min} - \min_{\substack{A' \subseteq \bigcup_{i=1}^{r} A_R(C_i) \\ |A'|=\alpha}} \sum_{a \in A'} (\overline{c}(a) - \underline{c}(a))(f(a) - \sum_{i \in I_a} \delta^i_{\min})$$

positiv sind. ◇

Beweis
$\Rightarrow$: Sei (f, B) ein optimaler (b, K)-Fluss des K-MKF-Problems mit Kosten $c(f)$ und $D_K(f)$ der entsprechende K-Residualgraph. Seien außerdem $C_1, \ldots, C_r$ Kreise in $D_K(f)$ mit den folgenden Eigenschaften:

- $A_R(C_i)$ ist die Menge aller K-Residualbogen des Kreises C_i,
- $A_K(C_i)$ ist die Menge aller K-Bogen des Kreises C_i,
- $\alpha = |\bigcup_{i=1}^{r} A_K(C_i)|$,
- I_a entspricht der Indexmenge aller Kreise C_i, die über einen Bogen $a \in A(D_K(f))$ verlaufen,
- $\delta^i_{\min}$ entspricht der Vielfachheit eines Kreises C_i, so dass für jeden Bogen $a \in A(D_K(f))$ gilt, dass $\sum_{i \in I_a} \delta^i_{\min} \leq u(a)$ ist,

und für diese Kreise gilt

$$|\bigcup_{i=1}^{r} A_K(C_i)| \leq |\bigcup_{i=1}^{r} A_R(C_i)|$$

und

$$\sum_{i=1}^{r} c(C_i)\delta^i_{\min} - \min_{\substack{A' \subseteq \bigcup_{i=1}^{r} A_R(C_i) \\ |A'|=\alpha}} \sum_{a \in A'} (\overline{c}(a) - \underline{c}(a))(f(a) - \sum_{i \in I_a} \delta^i_{\min}) > 0.$$

Sei außerdem

$$A_1 = \arg \min_{\substack{A' \subseteq \bigcup_{i=1}^{r} A_R(C_i) \\ |A'|=\alpha}} \sum_{a \in A'} (\overline{c}(a) - \underline{c}(a))(f(a) - \sum_{i \in I_a} \delta^i_{\min}).$$

Für einen Bogen $a \in A(D_K)$ bezeichnen wir den entsprechenden Bogen in $D_K(f)$ ebenfalls mit a und den entsprechenden Residualbogen mit $\vec{a}$.

Bilde nun einen (b, K)-Fluss (f', B') in dem Graphen D_K anhand des Flusses f und der Kreise $C_1, \ldots, C_r$ im K-Residualgraphen $D_K(f)$ wie folgt:
Für alle Bogen $a \in A(D_K)$ mit $f(a) > 0$ und $\vec{a} \in A_1$ entferne $\sum_{i \in I_a} \delta^i_{\min}$ Flusseinheiten von (f, B) von dem Bogen in D_K mit Kosten $\overline{c}(a)$ und verschicke die restlichen $f(a) - \sum_{i \in I_a} \delta^i_{min}$ Einheiten über den parallelen Bogen $a' \in A(D_K)$ mit Kosten $\underline{c}(a')$.
Für alle Bogen $a \in A(D_K)$ mit $f(a) > 0$ und $a \in \bigcup_{i=1}^{r} A_K(C_i)$ verschicke $\sum_{i \in I_a} \delta^i_{min}$ Flusseinheiten über den entsprechenden Bogen $a \in A(D_K)$ mit Kosten $\overline{c}(a)$.
Für alle neutralen Bogen $a \in A(D_K)$ mit $f(a) > 0$ und $a \in A(\bigcup_{i=1}^{r} C_i)$ verschicke $\sum_{i \in I_a} \delta^i_{min}$ Flusseinheiten über den entsprechenden Bogen $a \in A(D_K)$ mit Kosten $\underline{c}(a)$.
Für alle neutralen Bogen $a \in A(D_K)$ mit $f(a) > 0$ und dem neutralen Residualbogen $\vec{a} \in A(\bigcup_{i=1}^{r} C_i)$ entferne $\sum_{i \in I_a} \delta^i_{min}$ Flusseinheiten von dem entsprechenden Bogen $a \in A(D_K)$ mit Kosten $\underline{c}(a)$.
Für alle Bogen $a \in A(D_K)$ mit $f(a) > 0$ und $\vec{a} \in \bigcup_{i=1}^{r} \setminus \{A_1\}$ entferne $\sum_{i \in I_a} \delta^i_{\min}$ Flusseinheiten von (f, B) von dem Bogen in D_K mit Kosten $\overline{c}(a)$.
Alle Bogen $a \in A(D_K)$ mit $f(a) > 0$ und $\vec{a} \notin A(\bigcup_{i=1}^{r} C_i)$ bleiben unverändert.
Dies entspricht nun einem (b, K)-Fluss (f', B'). Da wir den Fluss (f, B) nur entlang von Kreisen im K-Residualgraphen $D_K(f)$ verändert haben. gilt weiterhin der Flusserhalt. Außerdem gilt, da (f, B) ein (b, K)-Fluss ist und die Kreise maximal soviel K-Bogen wie K-Residualbogen

verwendet haben, dass $|B'| \leq K$ ist. Für die Kosten dieses (b, K)-Flusses gilt, dass

$$\begin{aligned} c(f') &= c(f) + \sum_{i=1}^{r} c(C_i)\delta^i_{min} - \sum_{a \in A'} (\overline{c}(a) - \underline{c}(a))(f(a) - \sum_{i \in I_a} \delta^i_{min}) \\ &\geq c(f) + \sum_{i=1}^{r} c(C_i)\delta^i_{min} - \min_{\substack{A' \subset \bigcup_{i=1}^{r} A_R(C_i) \\ |A'| = \alpha}} \sum_{a \in A'} (\overline{c}(a) - \underline{c}(a))(f(a) - \sum_{i \in I_a} \delta^i_{\min}) \\ &> c(f) \end{aligned}$$

ist, mit $\alpha = |\bigcup_{i=1}^{r} A_K(C_i)|$. Dies ist jedoch ein Widerspruch dazu, dass (f, B) ein optimaler (b, K)-Fluss ist.

$\Leftarrow$: Zu zeigen ist noch, dass (f, B) ein optimaler (b, K)-Fluss ist, falls es keine solche Vereinigung von Kreisen gibt. Sei dazu (f, B) ein zulässiger (b, K)-Fluss in D_K, der aber nicht optimal ist. Es gibt in dem K-Graphen D_K somit mindestens einen weiteren zulässigen (b, K)-Fluss (f', B') mit $c(f') > c(f)$. Wir definieren nun eine Funktion $g' : A(D_K(f)) \to \mathbb{Z}^+$ in $D_K(f)$ und zeigen, dass dies eine *Zirkulation* ist. Für eine Zirkulation muss gelten, dass für alle Knoten $v \in D_K(f)$

$$\sum_{a \in \delta^+_{D_K(f)}(v)} g'(a) = \sum_{a \in \delta^-_{D_K(f)}(v)} g'(a)$$

ist. Sei $a \in A(D)$ ein Bogen mit zwei Kostenwerten $\overline{c}(a)$ und $\underline{c}(a)$. Dann sind a_1 und a_2 die dazugehörigen parallelen Bogen in $A(D_K)$. Der Hilfsbogen in $D_K(f)$ für die Bogen a_1 und a_2 wird mit a' bezeichnet und ein möglicher Residualbogen mit $\vec{a}$. Unsere Zirkulation definieren wir nun anhand der beiden b-Flüsse f und f' wie folgt:

Fall 1: Sei $f(a_1) > 0$ und $f'(a_2) > 0$. Dann ist $g'(a_2) = f'(a_2)$, $g'(a_1) = 0$, $g'(a') = f'(a_2)$ und $g'(\vec{a_1}) = f(a_1)$.

Fall 2: Sei $f(a_1) > 0$ und $f'(a_1) > 0$. Dann ist $g'(a_1) = \max\{f'(a_1) - f(a_1), 0\}$, $g'(a_2) = 0$, $g'(a') = \max\{f'(a_1) - f(a_1), 0\}$ und $g'(\vec{a_1}) = \max\{f(a_1) - f'(a_1), 0\}$.

Fall 3: Sei $f(a_2) > 0$ und $f'(a_2) > 0$. Dann ist $g'(a_2) = \max\{f'(a_2) - f(a_2), 0\}$, $g'(a_1) = 0$, $g'(a') = \max\{f'(a_2) - f(a_2), 0\}$ und $g'(\vec{a_2}) = \max\{f(a_2) - f'(a_2), 0\}$.

Fall 4: Sei $f(a_1) > 0$ und $f'(a_2) = f'(a_1) = 0$. Dann ist $g'(a_2) = g'(a_1) = 0$, $g'(a') = 0$ und $g'(\vec{a_1}) = f(a_1)$.

Fall 5: Sei $f(a_2) > 0$ und $f'(a_2) = f'(a_1) = 0$. Dann ist $g'(a_2) = g'(a_1) = 0$, $g'(a') = 0$ und $g'(\vec{a_2}) = f(a_2)$.

Fall 6: Sei $f'(a_1) > 0$ und $f(a_2) = f(a_1) = 0$. Dann ist $g'(a_1) = f'(a_1)$, $g'(a_2) = 0$ und $g'(a') = f'(a_1)$.

Fall 7: Sei $f'(a_2) > 0$ und $f(a_2) = f(a_1) = 0$. Dann ist $g'(a_2) = f'(a_2)$, $g'(a_1) = 0$ und $g'(a') = f'(a_2)$.

Zu zeigen ist, dass g' eine Zirkulation im K-Residualgraphen entspricht, also dass

$$\sum_{a \in \delta^+_{D_K(f)}(v)} g'(a) = \sum_{a \in \delta^-_{D_K(f)}(v)} g'(a)$$

für alle Knoten $v \in D_K(f)$ ist. Nach obiger Konstruktion gilt für einen Knoten $v \in A(D_K(f))$, dass

$$\sum_{a \in \delta^+_{D_K}(v)} f'(a) - \sum_{a \in \delta^+_{D_K}(v)} f(a) = \sum_{\substack{a \in \delta^+_{D_K(f)}(v): \\ a \in \delta^+_{D_K}(v)}} g'(a) - \sum_{\substack{a \in \delta^-_{D_K(f)}(v): \\ a \in \delta^+_{D_K}(v)}} g'(a)$$

ist und analog, dass

$$\sum_{a \in \delta^-_{D_K}(v)} f'(a) - \sum_{a \in \delta^-_{D_K}(v)} f(a) = \sum_{\substack{a \in \delta^-_{D_K(f)}(v): \\ a \in \delta^-_{D_K}(v)}} g'(a) - \sum_{\substack{a \in \delta^+_{D_K(f)}(v): \\ a \in \delta^-_{D_K}(v)}} g'(a)$$

ist. Damit ist

$$\begin{aligned}
& \sum_{a \in \delta^+_{D_K(f)}(v)} g'(a) - \sum_{a \in \delta^-_{D_K(f)}(v)} g'(a) \\
= & \sum_{\substack{a \in \delta^+_{D_K(f)}(v): \\ a \in \delta + -_{D_K}(v)}} g'(a) - \sum_{\substack{a \in \delta^-_{D_K(f)}(v): \\ a \in \delta^+_{D_K}(v)}} g'(a) - \sum_{\substack{a \in \delta^-_{D_K(f)}(v): \\ a \in \delta^-_{D_K}(v)}} g'(a) - \sum_{\substack{a \in \delta^-_{D_K(f)}(v): \\ a \in \delta^+_{D_K}(v)}} g'(a) \\
= & \sum_{a \in \delta^+_{D_K}(v)} f'(a) - f(a) - \sum_{a \in \delta^-_{D_K}(v)} f'(a) - f(a) \\
= & \sum_{a \in \delta^+_{D_K}(v)} f'(a) - \sum_{a \in \delta^-_{D_K}(v)} f'(a) + \sum_{a \in \delta^+_{D_K}(v)} f(a) - \sum_{a \in \delta^-_{D_K}(v)} f(a) \\
= & \; b(v) - b(v) \\
= & \; 0
\end{aligned}$$

für alle Knoten $v \in D_K(f)$. Also ist g' eine Zirkulation in $D_K(f)$. Die Kreise, die dieser Zirkulation entsprechen, bezeichnen wir mit $C_1, \ldots, C_r$. Diese Kreise besitzen, (f, B) und (f', B') zulässige (b, K)-Flüsse sind, maximal K K-Residualbogen und maximal soviele K-Bogen wie K-Residualbogen.. Jeder Kreis C_i besitzt eine Kapazität von $\delta^i_{\min} = \min\limits_{a \in A(C_i)} g'(a)$, womit insgesamt für einen Bogen $a \in A(D_K(f))$ gilt, dass $\sum\limits_{i \in I_a} \delta^i_{\min} \leq u(a)$ gilt. Außerdem gilt für diese Kreise

$$
\begin{aligned}
& \sum_{i=1}^{r} c(C_i)\delta^i_{min} - \min_{\substack{A' \subset \bigcup_{i=1}^{r} A_R(C_i) \\ |A'|=K}} \sum_{a \in A'} (\overline{c}(a) - \underline{c}(a))(f(a) - \sum_{i \in I_a} \delta^i_{\min}) \\
= & \sum_{i=1}^{r} c(C_i)\delta^i_{min} - \sum_{a \in \vec{B}} (\overline{c}(a) - \underline{c}(a))(f(a) - \sum_{i \in I_a} \delta^i_{\min}) \\
= & \sum_{i=1}^{r} c(C_i)\delta^i_{min} \\
= & -c(f) + c(f') \\
> & \ 0
\end{aligned}
$$

mit $A_R(C_i)$ der Menge der K-Residualbogen in einem Kreis C_i. Also gilt, dass falls (f, B) kein optimaler (b, K)-Fluss in D_K ist, es eine wie oben beschriebene Kreiszerlegung gibt. ■

Eine solche Kreisvereinigung wie die Zirkulation aus Optimalitätskriterium 5.15, bezeichnen wir folgend als *K-Kreisvereinigung.*
Mithilfe des Optimalitätskriteriums aus Satz 5.15 kann man Algorithmen entwickeln, wie man ausgehend von einem zulässigen (b, K)-Fluss (f, B) einen optimalen (b, K)-Fluss finden kann. Falls die Suche nach einer solchen Zirkulation in polynomieller Zeit durchgeführt werden kann, kann mithilfe des Optimalitätskriteriums 5.15 ein pseudopolynomieller Algorithmus zur Bestimmung des optimalen (b, K)-Flusses entwickelt werden. Die Suche nach einer solchen Kreisvereinigung $W = C_1 \cup \ldots \cup C_r$ muss jedoch nicht ohne weiteres durchführbar sein, wie der folgende Satz zeigt.

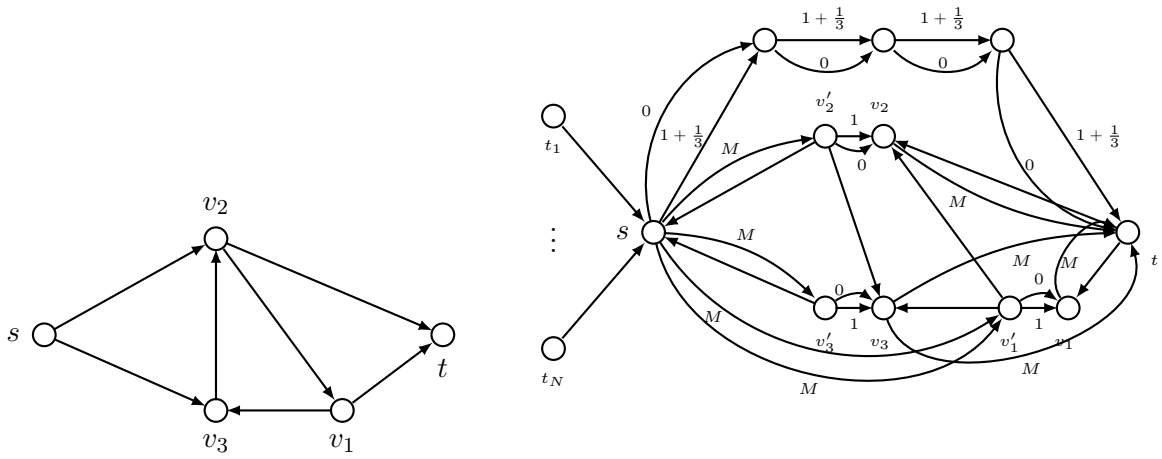
(5.16) Satz
Sei $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, u, b, K)$ eine Instanz des K-MKF-Problems. Das Finden einer K-Kreisvereinigung $W = C_1 \cup \ldots \cup C_r$ mit

$$
\sum_{i=1}^{r} c(C_i)\delta^i_{\min} - \min_{\substack{A' \subseteq \bigcup_{i=1}^{r} A_R(C_i) \\ |A'|=\alpha}} \sum_{a \in A'} (\overline{c}(a) - \underline{c}(a))(f(a) - \sum_{i \in I_a} \delta^i_{\min}) > 0
$$

in einem K-Residualgraphen $D_K(f)$ zu einem zulässigen (b, K)-Fluss (f, B) ist stark $\mathcal{NP}$-vollständig. ◇

Beweis
Das Überprüfen, ob eine gegebene Lösung eine zulässige K-Kreisvereinigung mit den gesuchten Kosten ist, ist in polynomieller Zeit durchführbar, daher ist das Problem in $\mathcal{NP}$.
Wir wollen diesen Satz mit einer Reduktion des gerichteten Hamiltonweg-Problems zeigen. Sei $D = (V, A)$ ein beliebiger Digraph mit Quelle $s \in V(D)$ und Senke $t \in V(D)$. Wir wollen diesen Graphen D zu einem Graphen D' transformieren. Zunächst kehren wir dazu die Orientierung aller Bogen von D um. Dadurch ändern sich die Quelle-Senke-Eigenschaften. Wir wählen eine beliebige Anordnung der Knotenmenge $V(D)$ und benutzen diese als topologische Sortierung. Für eine solche Anordnung können die Quelle-Senke-Verhältnisse verwendet werden. Die Orientierung der Bogen $a \in A(D)$, die dieser topologischen Anordnung widersprechen, wird wieder umgekehrt.
Wir erstellen für jeden Knoten $v \in V(D)\backslash\{s, t\}$ eine Kopie in D' und benenne diese jeweils mit v'. Alle ausgehenden Bogen $vw \in A(D)$ werden zu Bogen $v'w$ in $A(D')$ umgewandelt. Diese Bogen erhalten in D' Kosten von 0 und eine Kapazität von unendlich. Anschließend fügen wir je zwei parallele Bogen $v'v$ für alle $v \in V(D)$ ein, wobei einer dieser parallelen Bogen Kosten von 0 und eine Kapazität von 1 erhält und der andere Bogen Kosten von 1 und eine Kapazität von ebenfalls 1.
Anschließend fügen wir dem Graphen D' noch $|V(D)| - 3$ weitere Knoten k_i hinzu und jeweils zwei parallele Bogen $sk_1, k_1k_2, \ldots, k_ik_{i+1}, \ldots, k_{|V(D)|-3}t$. Einer dieser parallelen Bogen erhält jeweils Kosten von $1 + \frac{1}{|V(D)|-2}$ und eine Kapazität von 1, der andere Kosten 0 und ebenfalls eine Kapazität von 1. Für jeden Bogen $u'w \in A(D')$ mit $uw \in A(D)$ und jedes Knotenpaar v', v mit $v \in V(D)$ werden nun noch weitere Bogen dem Graphen D' hinzugefügt, nämlich die Bogen su beziehungsweise sv' sowie die Bogen wt beziehungsweise vt. Diese Bogen erhalten alle Kosten von $|V(D)||A(D)|$ und eine Kapazität von 1. Außerdem werden $|V(D)| - 2 + |A'(D)|$ weitere Knoten $t_1, \ldots, t_N$ mit $N = |V(D)| - 2 + |A'(D)|$ eingefügt, wobei $|A'(D)|$ die Anzahl an Bogen, die entgegen der topologischen Sortierung verliefen, angibt. Für jeden dieser Knoten t_i wird ein st_i-Bogen erstellt mit Kosten 0 und einer Kapazität von 1.
Für $t \in V(D')$ definieren wir die Balance $|V(D)| - 2 + |A'(D)|$ und für jedes t_i mit $i = 1, \ldots, N$ die Balance -1.

In Abbildung 5.8 sieht man eine solche Transformation eines Graphen D in Abbildung 5.8a zu dem Graphen D' in der Abbildung 5.8b. Es wurde hier die Sortierung t, v_1, v_2, v_3, s gewählt. Zur Vereinfachung der Kosten $|V(G)||A(G)|$ wird hier nur Wert $M := |V(G)||A(G)|$ angegeben. Nicht ausgezeichnete Bogen in Abbildung 5.8b haben Kosten von 0. Eine solche Transformation ist in polynomieller Zeit durchführbar, da $3|V(D)| - 2$ Knoten erzeugt werden und maximal $3|A(D)| + |V(D)| - 1$ Bogen. Die Bogen $sk_1, k_1k_2, \ldots, k_ik_{i+1}, \ldots, k_{|V(D)|-3}t$ mit den jeweiligen Kosten $1 + \frac{1}{|V(|D|)-2}$ sowie die Bogen $v'v$ mit $v \in V(D)$ und Kosten von 1 entsprechen K-Bogen und alle übrigen Bogen entsprechen neutralen Bogen.



(a) Ursprünglicher Graph D (b) Transformierter Graph D'

Abbildung 5.8: Reduktion 1: Transformation

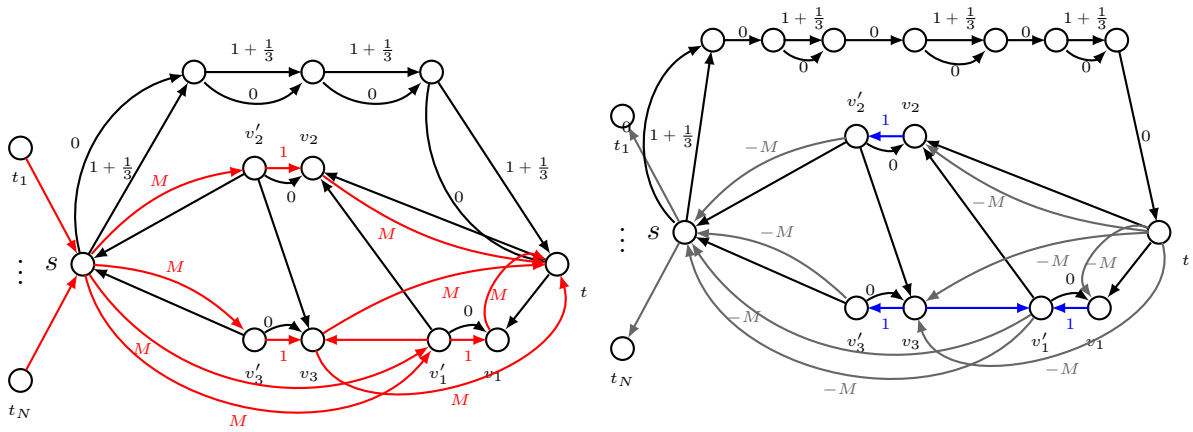
Der Graph D' entspricht einem K-Graphen. Diesen belegen wir mit einem (b, K)-Fluss (f, B), um einen K-Residualgraphen von D' zu erhalten. K ist hierbei $|V(D)| - 2$. Dafür schicken wir für alle Knoten $v \in V(D)$ jeweils eine Einheit über den Weg $t_i svv't$ für ein $i = 1, \ldots, N$, sowie für alle Bogen uw, die entgegen der gewählten Linearisierung verliefen, jeweils eine Einheit über den Weg $t_i swu't$ mit $i = 1, \ldots, N$. Jeder dieser Wege verwendet dabei eine andere Senke t_i mit $i = 1, \ldots, N$. Die Menge B ist die Menge $\{vv' | v \in V(D)\}$. Dies entspricht also einem (b, K)-Fluss (f, B). Es werden maximal $|V(D)| + |A(D)|$ Einheiten versandt.

Anschließend wird der K-Residualgraphen $D_K(f)$ bezüglich dieses Flusses (f, B) gebildet. Der Fluss, der zu dem transformierten Graphen aus Abbildung 5.8 gehört, und der entsprechende K-Residualgraph sind nun der Abbildung 5.9 zu entnehmen. Der gewählte Fluss wird dabei in der Teilabbildung 5.9a wie zuvor durch die roten Bogen dargestellt. In der Abbildung 5.9b ist der dazugehörige K-Residualgraph zu sehen. Die Bogen mit Kosten von M sind grau markiert und in blau sind die übrigen Residualbogen zu sehen.

Wir zeigen nun, dass es in D genau dann einen gerichteten Hamiltonweg gibt, wenn es in dem Residualgraphen $D_K(f)$ des Graphen D' zum Fluss f eine gewünschte Kreisvereinigung W mit

$$\sum_{i=1}^{r} c(C_i)\delta^i_{\min} - \min_{\substack{A' \subseteq \bigcup_{i=1}^{r} A_R(C_i) \\ |A'|=\alpha}} \sum_{a \in A'} (\overline{c}(a) - \underline{c}(a))(f(a) - \sum_{i \in I_a} \delta^i_{\min}) > 0$$

gibt.



(a) (b, K)-Fluss (f, B) im Graphen D_K (b) Residualgraph $D_K(f)$

Abbildung 5.9: Reduktion 1: Fluss f und K-Residualgraph $D_K(f)$

$\Rightarrow$: Sei $P_1 = sw_1w_2 \dots w_{|V(D)|-2}t$ ein Hamiltonweg in D. Dann entspricht der Kreis $C = P_1 \cup P_2$, wobei P_2 der verbleibende s-t-Weg in $D_K(f)$ ist, bei dem für zwei parallele Bogen der Bogen mit Kosten von $1 + \frac{1}{|V(D)|-2}$ verwendet wird, einem Kreis in $D_K(f)$, der über $|V(D)| - 2$ K-Bogen sowie $|V(D)| - 2$ K-Residualbogen verläuft, einfach ist und insgesamt Kosten von 1 besitzt. Also ist C als einzelner Kreis eine K-Kreisvereinigung mit $r = 1$ und

$$\begin{aligned} & \sum_{i=1}^{r} c(C_i)\delta^i_{\min} - \min_{\substack{A' \subseteq \bigcup_{i=1}^{r} A_R(C_i) \\ |A'|=\alpha}} \sum_{a \in A'} (\overline{c}(a) - \underline{c}(a))(f(a) - \sum_{i \in I_a} \delta^i_{\min}) \\ = \; & c(C) \\ = \; & 1 \\ > \; & 0. \end{aligned}$$

$\Leftarrow$: Sei W eine positive K-Kreisvereinigung in $D_K(f)$ mit

$$\sum_{i=1}^{r} c(C_i)\delta^i_{\min} - \min_{\substack{A' \subseteq \bigcup_{i=1}^{r} A_R(C_i) \\ |A'|=\alpha}} \sum_{a \in A'} (\overline{c}(a) - \underline{c}(a))(f(a) - \sum_{i \in I_a} \delta^i_{\min}) > 0.$$

Da in $D_K(f)$ nur die Bogen $sk_1, k_1k_2, \dots, k_ik_{i+1}, \dots, k_{|V(G)|-3}t$ positive Kosten besitzen sind, läuft C also über den s-t-Weg P in $D_K(f)$, der diese Bogen mit Kosten $1 + \frac{1}{|V(D)|-3}$ verwendet. Damit beinhaltet C also $|V(D)| - 2$ K-Bogen. Da der Graph $D_K(f)$ ohne den Weg P azyklisch

ist, besteht die K-Kreisvereinigung nur aus einem Kreis. In $D_K(f)\backslash\{sk_1, k_1k_2, \ldots, k_ik_{i+1}, \ldots, v_{|V(D)|-3}t\}$ müssen mindestens $|V(D)| - 2$ K-Residualbogen auf diesem einen Kreis C enthalten sein, da C sonst keine K-Kreisvereinigung ist. Alle K-Residualbogen besitzen Kosten von -1, weswegen jeder genau einmal durchlaufen werden muss. Der Kreis C ist bogendisjunkt, also einfach, und damit entspricht $C\backslash P$ einem gerichteten Hamiltonweg in $D_K(f)\backslash P$ und die entsprechenden Bogen im Graphen D auch einem gerichteten Hamiltonweg in D. ■

Also ist das Finden einer einfachen K-Kreisvereinigung $\mathcal{NP}$-vollständig. Das führt uns zu folgender Vermutung.

(5.17) Vermutung
Das K-MKF-Problem ist $\mathcal{NP}$-vollständig. ◇

6 Lösungsverfahren für das K-MKF-Problem

In Kapitel 5 haben wir das K-MKF-Problem genauer betrachtet und dabei für den Spezialfall von azyklischen Digraphen mit einer Senke polynomielle Algorithmen kennengelernt. Für den allgemeinen Fall mit $n \geq 2$ Senken betrachteten wir ein Optimalitätskriterium, was vermuten lässt, dass das K-MKF-Problem $\mathcal{NP}$-vollständig ist.
Bislang haben wir als Lösungsverfahren einer Instanz des K-MKF-Problems im allgemeinen Fall nur das gemischt-ganzzahlige lineare Programm, wiederholt durch Modell 19, kennengelernt. In diesem Kapitel wollen wir uns mit weiteren Lösungsverfahren des K-MKF-Problems beschäftigen. Da wir annehmen, dass das K-MKF-Problem $\mathcal{NP}$- vollständig ist, wird es vermutlich, sofern $\mathcal{P} \neq \mathcal{NP}$ gilt, keinen polynomiellen Algorithmus zur Lösung des K-MKF-Problems geben.

Modell 19 Ganzzahliges lineares Programm des K-MKF-Problems

$$
\begin{aligned}
\max \quad & \sum_{a \in A(D)} \underline{c}(a) y(a) + \sum_{a \in A(D)} (\overline{c}(a) - \underline{c}(a)) w(a) & \\
s.t. \quad & \sum_{a \in \delta^+(v)} y(a) - \sum_{a \in \delta^-(v)} y(a) = b(v) & \forall v \in V(D) \\
& \sum_{a \in A(D)} z(a) \leq K & \\
& z(a) \leq y(a) & \forall a \in A(D) \\
& w(a) \leq M z(a) & \forall a \in A(D) \\
& w(a) \leq y(a) & \forall a \in A(D) \\
& z(a) \in \{0, 1\} & \forall a \in A(D) \\
& w(a) \in \mathbb{Z}^+ & \forall a \in A(D) \\
& y(a) \in \mathbb{Z}^+ & \forall a \in A(D)
\end{aligned}
$$

Falls für einige Bogen $a \in A(D)$ $\overline{c}(a) = \underline{c}(a)$ gilt, so ergänze man die Nebenbedingung $z(a) = 1$ für diese Bogen in das Modell 19. Um in polynomieller Zeit Lösungen des K-MKF-Problems zu finden, kann man etwa Heuristiken verwenden, jedoch wollen wir uns in dieser Arbeit auf exakte Verfahren fokussieren. Die in diesem Kapitel betrachteten Lösungsverfahren sind einmal eine Lösungsstrategie über eine Ausnutzung der (b, K)-Flusseigenschaften und zum anderen ein Column Generation Verfahren. In Kapitel 4 sahen wir bereits das allgemeine Lösungsschema für die robuste Terminvergabe im Krankenhaus unter unsicheren Behandlungspfaden, jedoch sind wir an dieser Stelle nicht näher auf Lösungsverfahren für das K-MKF-Problems eingegangen. Die beiden in diesem Kapitel vorgestellten Verfahren sowie das Lösen als ganzzahliges lineares Programm eignen sich allesamt als Lösungsverfahren für das K-MKF-Problem während der robusten Terminvergabe.

Wir verzichten in diesem Kapitel auf das Betrachten des K-MKF-Problems unter Kapazitätsbedingungen, da in unserer betrachteten Problemstellung der robusten Terminvergabe im Krankenhaus keine solchen gegeben sind und sich alle hier in diesem Kapitel vorgestellten Verfahren für Bogenkapazitäten verallgemeinern lassen.

6.1 Das Trennungspunkteverfahren

Wir wollen nun ein erstes neues Lösungsverfahren betrachten. Dieses basiert auf einer genaueren Betrachtung und Ausnutzung der Eigenschaften eines (b, K)-Flusses. Sei dazu $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ eine Instanz des K-MKF-Problems ohne Kapazitätsbedingungen. Für eine optimale Lösung dieser Instanz wird ein (b, K)-Fluss (f, B) gesucht. Der Weg, den eine Flusseinheit von f im Graphen D von s zu einer Senke t_i zurücklegt, entspricht einem s-t_i-Weg mit $i \in \{1, \ldots, n\}$. Alle diese Wege des (b, K)-Flusses starten beim gleichen Knoten, nämlich bei der Quelle s. Die Wege sind also nicht knotendisjunkt. Die Quelle s muss jedoch nicht der letzte Knoten gewesen sein, den zwei Wege des (b, K)-Flusses teilen.

(6.1) Definition (Trennungspunkt)
Sei $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ eine Instanz eines K-MKF-Problems und (f, B) ein zulässiger (b, K)-Fluss. Seien $P_1, \ldots, P_n$ die Wege von der Quelle s zu den Senken $t_1, \ldots, t_n$ des b-Flusses f. Ein **Trennungspunkt** ist ein Knoten $v \in V(D)$ für den für zwei Wege P_i und P_j mit $P_i \neq P_j$ gilt, dass $v \in P_i$ und $v \in P_j$ ist und die Nachfolgeknoten von v auf den Wegen P_i und P_j verschieden sind. Für den Nachfolger v_{i+1} von v auf P_i und den Nachfolger v_{j+1} auf P_j gilt, dass $v_{i+1} \neq v_{j+1}$ ist. ◇

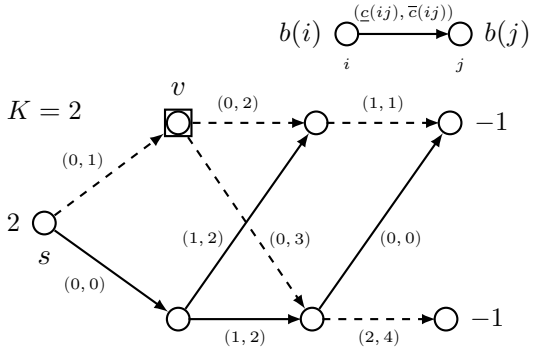


Abbildung 6.1: Trennungspunkte

Offensichtlich existieren für je zwei Wege der zulässige Lösungen des K-MKF-Problems Trennungspunkte, da alle Wege bei demselben Knoten starten, aber zu verschiedenen Senken führen. In Abbildung 6.1 ist der b-Fluss f eines zulässigen (b, K)-Flusses (f, B) durch die gestrichelten Bogen in einem azyklischen Digraphen mit zwei Senken zu sehen, wobei der markierte

Knoten v der Trennungspunkt der beiden Quelle-Senke-Wege ist. Welche Bogen der Menge B entsprechen ist hier nicht weiter von Bedeutung.

Allgemein können für zwei Wege P_i und P_j auch Verbindungspunkte bestehen.

(6.2) Definition (Verbindungspunkt)
Sei $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ eine Instanz eines K-MKF-Problems und (f, B) ein zulässiger (b, K)-Fluss. Seien $P_1, \ldots, P_n$ die Wege von der Quelle s zu den Senken $t_1, \ldots, t_n$ des b-Flusses f. Ein **Verbindungspunkt** ist ein Knoten $v \in V(D)$ für den für zwei Wege P_i und P_j mit $P_i \neq P_j$ gilt, dass $v \in P_i$ und $v \in P_j$ ist, aber die Vorgängerknoten von v auf den Wegen P_i und P_j verschieden sind. Für den Vorgänger v_{i-1} von v auf P_i und den Vorgänger v_{j-1} auf P_j gilt also $v_{i-1} \neq v_{j-1}$. ◇

Verbindungspunkte treten allerdings im Falle einer optimalen Lösung (f, B) des K-MKF-Problems nicht auf, wie der folgende Satz aussagt.

(6.3) Satz
In einer optimalen Lösung (f, B) einer Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ des K-MKF-Problems existieren keine Verbindungspunkte zwischen den s-t_i-Wegen P_i von f für $i = 1, \ldots, n$. ◇

Beweis
Sei (f, B) eine optimale Lösung des K-MKF-Problems der Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ und P_i für $i = 1, \ldots, n$ der entsprechende s-t_i-Weg des b-Flusses f. Angenommen, es existieren zwei Wege P_i und P_j mit $i \neq j$, sodass es einen Trennungspunkt v, aber auch einen Verbindungspunkt w mit $w \neq s$ gibt. Der Verbindungspunkt w tritt auf beiden Wegen nach dem Trennungspunkt auf, da beide Wege bei s starten. Es muss dann mindestens einen weiteren Trennungspunkt v' geben, da die Wege zu zwei verschiedenen Senken führen. Angenommen, $c(P_{i_{|[v,w]}}) < c(P_{j_{|[v,w]}})$. Dann ist der Fluss f', bei dem der Weg P_i durch den Weg $P_{j_{|[s,v']}} \cup P_{i_{|(v',t_i]}}$ ausgetauscht wird, ein Fluss mit $c(f') > c(f)$, was ein Widerspruch zu der Annahme ist, dass (f, B) ein optimaler (b, K)-Fluss ist. Also können in optimalen Lösungen des K-MKF-Problems keine Verbindungspunkte auftreten. ■

(6.4) Folgerung
In einer optimalen Lösung des K-MKF-Problems in azyklischen Digraphen mit n Senken kann es maximal $n - 1$ Trennungspunkte geben. ◇

Insbesondere der Quellknoten s kann ein möglicher Trennungspunkt sein. Wenn an einem Trennungspunkt mehr als zwei Wege getrennt werden, sind es insgesamt in dem betrachteten Digraphen D weniger als $n - 1$ Trennungspunkte, sofern jeder nur einfach gezählt wird.

(6.5) Satz
Sei eine Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ des K-MKF-Problems gegeben. Falls die Menge $\mathcal{T}$ aller Trennungspunkte einer optimalen Lösung f des K-MKF-Problems gegeben ist und $\mathcal{T}' := \mathcal{T} \cup \{s, t_1, \ldots, t_n\}$ die Menge aller Trennungspunkte, aller Senken und der Quelle s ist, dann definiert eine optimale Lösung des K-MKF-Problems einen aufspannenden Wurzelbaum auf den Knoten der Menge $\mathcal{T}'$ mit Wurzel s. ◇

Beweis
Eine optimale Lösung (f, B) des K-MKF-Problems enthält alle s-t_i-Wege des b-Flusses f für $i = 1, \ldots, n$. Ein Trennungspunkt $T \in \mathcal{T}$ trennt nun mindestens zwei dieser Wege P_i, P_j mit $i \neq j$. Der zugrunde liegende Digraph D ist azyklisch und besitzt keine Verbindungspunkte für den Fluss f, daher entsprechen die Bogen, über die in einer optimalen Lösung (f, B) Fluss fließt, einem Wurzelbaum mit s als Wurzel und den Senken $t_1, \ldots, t_n$ als Blätter. An jedem Trennungspunkt verzweigt sich der Fluss in mindestens zwei Kinder, da hier mindestens zwei Wege getrennt werden. D ist azyklisch und da der Fluss f einen Wurzelbaum aufspannt, existiert zwischen je zwei Knoten v und w aus der Menge $\mathcal{T}'$ ein eindeutiger Weg. Daher entspricht eine optimale Lösung einem aufspannenden Wurzelbaum auf den Knoten der Menge $\mathcal{T}' = \mathcal{T} \cup \{s, t_1, \ldots, t_n\}$. ■

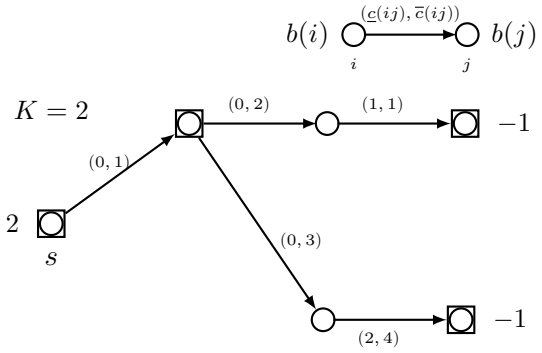


Abbildung 6.2: Darstellung des Spannbaums definiert über die Trennungspunkte

In Abbildung 6.2 sehen wir anhand des (b, K)-Flusses aus Abbildung 6.1, wie ein solcher aufspannender Wurzelbaum, kurz Spannbaum, aussehen kann. Die Knoten der Menge $\mathcal{T}'$ sind hier durch die Quadrate markiert worden und alle Knoten, die nicht auf dem Spannbaum liegen, wurden zur Veranschauung vernachlässigt.

Wenn nur die optimalen Trennungspunkte einer Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ des K-MKF-Problems gegeben sind, ist noch nicht klar, welche Wege des b-Flusses in einer optimalen Lösung an dieser Stelle getrennt werden. Aufgrund von Satz 6.5 wissen wir, dass ein aufspannender Baum auf diesen Trennungspunkten gesucht ist. Falls dieser Spannbaum gegeben ist, wird zwischen je zwei Trennungspunkten in einer optimalen Lösung ein eindeutiger Weg existieren, auf dem für $k \leq K$ Bogen die Kosten von $\underline{c}$ auf $\overline{c}$ erhöht werden. Unklar ist jedoch auch noch, wie viele Bogenkosten auf einem Abschnitt zwischen zwei Trennungspunkten mit Kosten $\overline{c}$ gewählt werden dürfen. Sicher ist dabei nur, dass insgesamt auf dem gesamten Spannbaum maximal K Bogen mit diesen Kostenwerten gewählt werden dürfen.

Das Verfahren, dass wir in diesem Abschnitt betrachten wollen, ist ein Vorgehen, bei dem die soeben beschriebenen Probleme nach und nach gelöst werden. Dieses Vorgehen wird in Algorithmus 6 schematisch dargestellt.

Algorithmus 6 Trennungspunkteverfahren

Input: eine Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ eines K-MKF-Problems
Output: ein kostenmaximaler (b, K)-Fluss f
begin
  while *Noch nicht alle Möglichkeiten betrachtet worden sind* **do**
    Wähle eine mögliche Menge $\mathcal{T}$ der Trennungspunkte;
    Sei $\mathcal{T}' := \mathcal{T} \cup \{s, t_1, \ldots, t_n\}$;
    Wähle einen Spannbaum auf der Menge der Trennungspunkte;
    for $k \leq K$ **do**
      Berechne für jeden Abschnitt zwischen zwei Trennungspunkten eine Lösung des k-MKF-Problems in D mit $n = 1$;
    end
    Berechne eine optimale Verteilung der k zu erhöhenden Bogen auf allen Abschnitten zwischen den Trennungspunkten;
  end
  return *ein kostenmaximaler (b, K)-Fluss anhand eines optimalen Spannbaums und k;*
end

(6.6) Definition (Trennungspunktegraph $G_{\mathcal{T}}$)
Sei eine Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ des K-MKF-Problems und die Menge $\mathcal{T}$ an optimalen Trennungspunkten gegeben und sei $\mathcal{T}' := \mathcal{T} \cup \{s, t_1, \ldots, t_n\}$. Sei $G(\mathcal{T})$ der *Trennungspunktegraph* mit $V(\mathcal{T}) = \mathcal{T}'$. Für alle Knoten u und v in $\mathcal{T}'$ existiert in $G(\mathcal{T})$ ein Bogen uv, falls zwischen den dazugehörigen Trennungspunkten u und v in D ein gerichteter u-v-Weg existiert. Dieser Graph ist azyklisch, aber im Allgemeinen kein Baum. ◇

Ein solcher Trennungspunktegraph anhand der Abbildung 6.1 wird in der folgenden Abbildung 6.3 präsentiert. Da in linearer Zeit geprüft werden kann, ob ein Weg zwischen zwei Knoten in azyklischen Digraphen existiert, ist die Erstellung des Trennungspunktegraphen in polynomieller Zeit möglich. Der Trennungspunktegraph ist eine vereinfachte Anschauung der überhaupt für die Suche benötigten Wege. Diese Struktur benötigen wir später, um die weiteren Schritte zu verfolgen.

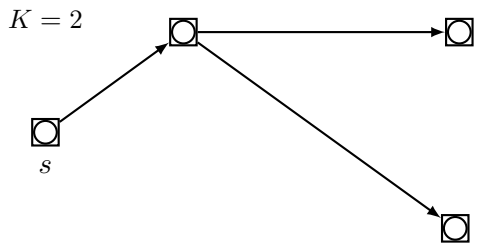


Abbildung 6.3: Der Trennungspunktegraph $G_{\mathcal{T}}$

Auf diesem Trennungspunktegraphen $G_{\mathcal{T}}$ suchen wir nun, sofern dieser keinem Baum entspricht, den gerichteten Spannbaum, der durch die optimale Lösung gegeben wird. Dazu müssen wir alle Spannbäume betrachten und untersuchen. Für einen gegebenen Digraphen $D = (V, A)$ lässt sich in $\mathcal{O}(|V(D)| + |A(D)|)$ ein minimaler Spannbaum berechnen. In Kapoor und Ramesh [14] wird beschrieben, wie man mit einem Aufwand von $\mathcal{O}(N|V(D)| + |V(D)|^3)$ alle gerichteten Spannbäume eines Digraphen D berechnen kann, wobei N die Anzahl aller möglichen Spannbäume ist. Für eine exponentielle Anzahl an Spannbäumen ist dies also auch ein exponentieller Algorithmus. Den Algorithmus, der in Kapoor und Ramesh [14] vorgestellt wird, wollen wir hier jedoch nicht vorstellen. Für jeden Trennungspunktegraphen müssen wir nun alle möglichen Spannbäume berechnen. Einen solchen Spannbaum von $G_{\mathcal{T}}$ bezeichnen wir auch mit T_S.

In einem Spannbaum T_S gibt es zwischen je zwei Knoten v und w einen eindeutigen Weg P. Anhand dieses Weges P lässt sich ermitteln, welche s-t_i-Wege in dem Graphen D der Lösung des K-MKF-Problems entsprechen, denn die s-t_i-Wege im Spannbaum T_S entsprechen Wegen P_i der optimalen Lösung f. Auf all diesen Wegen sollen insgesamt maximal K Bogen erhöht werden, damit die Kosten maximal sind. Unklar ist allerdings, wie viele Bogen auf den v-w-Wegen in D für alle Knotenpaare $vw \in \mathcal{T}' = V(T_S)$ erhöht werden sollen. Um dies berechnen zu können, treffen wir eine Veränderung an dem Spannbaum T_S.

Wir berechnen für jeden Bogen $a = vw \in A(T_S)$ und für jedes $k \leq K$ in dem Ursprungsgraphen D eine optimale Lösung des K-MKF-Problemsmit einer Quelle v und Senke w. Mithilfe von Kapitel 5 und für ein festes k das Maximalkostenflussproblem mit k variablen Kosten kann ein solches Problem in $\mathcal{O}(|V(D)|^3)$ gelöst werden. Da $K \leq |V(D)|$ gilt, ergibt sich für die Berechnung aller dieser Maximalkostenflussprobleme eine Komplexität von $\mathcal{O}(|A(D)||V(D)|^4) = \mathcal{O}(|V(D)|^6)$. Den Bogen $a \in A(T_S)$ ersetzen wir nun in dem Spannbaum T_S durch $K + 1$ parallele Bogen $a_0, \ldots, a_K$, die jeweils genau einem Wert $k \leq K$ entsprechen und die Kosten der optimalen Lösung des k-MKF-Problems zwischen den Trennpunkten, die den Bogen $a \in A(T_S)$ definiert, besitzen. Jeder Bogen entspricht nun also einer Lösung des k-MKF-Problems für $k \leq K$ mit Quelle $s \in V(D)$ und Senke $w \in V(D)$. Den veränderten Wurzelspannbaum T_S bezeichnen wir nun mit T'_S.

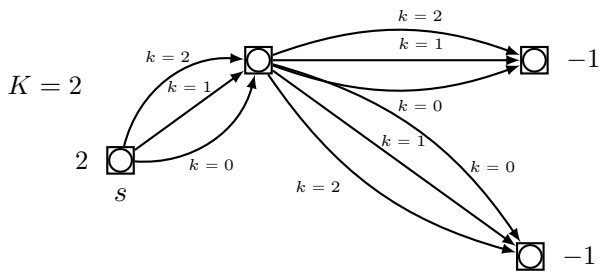


Abbildung 6.4: Angepasster Wurzelspannbaum T'_S

Abbildung 6.4 zeigt den angepassten Spannbaum T'_S. Mithilfe des folgenden Dynamischen Programms 7 kann nun eine optimale Verteilung der Anzahl der zu erhöhenden Bogen auf dem angepassten Spannbaum T'_S berechnet werden. Dazu sei

$P(v,k)$: die maximalen Kosten, die in dem Teilbaum mit $v \in V(T'_S)$ als Wurzelknoten mit k erhöhten Bogenkosten zu erzielen sind

$f(v,w,k)$: die Anzahl an Bogen, die zwischen $vw \in A(T'_S)$ erhöht werden, wenn ab Knoten v insgesamt k Bogenkosten erhöht werden

N_v : die Menge aller Knoten $w \in V(T'_S)$ mit $vw \in A(T'_S)$

Der Teilbaum mit v als Wurzelknoten meint den aufspannenden Teilgraphen, der durch den Wurzelknoten v und alle seine Nachfolgeknoten aufgespannt wird. Für jeden Bogen $vw \in A(T'_S)$ beschreibt vw_k den Bogen $vw \in A(T'_S)$, auf dessen entsprechendem Teilstück im Digraphen D k Bogenkosten mit Kosten von $\overline{c}$ verwendet werden dürfen, wobei $k \leq K$ ist. Außerdem ist es wichtig zu wissen, dass ein solcher angepasster Spannbaum T'_S zwar nicht mehr schlicht, aber dennoch azyklisch ist. Das bedeutet, dass man eine topologische Sortierung $v_1, \ldots, v_{|V(T'_S)|}$ der Knoten finden kann.

Das Dynamische Programm ist damit das Folgende.

Algorithmus 7 DP zur Berechnung einer optimalen Anzahl zu erhöhender Bogenkosten

Input: Ein angepasster Wurzelspannbaum T'_S auf der Menge der Trennungspunkte $\mathcal{T}'$ einer Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ des K-MKF-Problems

Output: Eine optimale Verteilung, wie viele Bogenkosten auf einem v-w-Weg in D für alle $vw \in A(T'_S)$ erhöht werden sollen und die optimalen Kosten dieser Verteilung

begin

  for $i = 1, \ldots, n$ **do**

    for $0 \leq k \leq K$ **do**

      $P(t_i, k) = 0$;

    end

  end

  forall $v = v_{|V(T'_S)|-1}, \ldots, v_1$ **do**

    for $1 \leq k \leq K$ **do**

      $P(v,k) = \max\{ \sum_{w \in N_v} \{P(w, k_w^1) + c(vw_{k_w^2}) | \sum_{w \in N_v} k_w^1 + k_w^2 = k\}$;

      $f(v,w,k) = \arg\max_{k_w^2} \{ \sum_{u \in N_v} P(u, k_u^1) + c(vu_{k_u^2})\}$;

    end

  end

  return $OPT = \max_{k \leq K}\{P(s,k)\}$ *und* f;

end

Das Dynamische Programm 7 liefert für einen angepassten Spannbaum T'_S eine optimale Lösung in polynomieller Zeit, falls der maximale Ausgangsgrad des Spannbaums T beschränkt ist, wie der folgende Satz 6.7 zeigt. Sei dazu $|d^+(v)| \leq d$ für alle Knoten $v \in V(T)$ mit $d \in \mathbb{N}$.

(6.7) Satz
Das Dynamische Programm 7 liefert eine optimale Lösung des K-MKF-Problems in polynomieller Zeit, falls der maximale Ausgangsgrad des gegebenen Spannbaums T durch eine Konstante $d \in \mathbb{N}$ beschränkt ist. ◇

Beweis
Das Dynamische Programm 7 liefert eine korrekte Lösung des K-MKF-Problems für einen angepassten Spannbaum T'_S:

Die Korrektheit zeigen wir per Induktion.

Induktionsanfang: Offensichtlich ist $P(t_i, r) = 0$ für alle $i = 1, \ldots, n$ sowie $r = 0, \ldots, K$, da von jeder Senke t_i keine Bogen abgehen.

Induktionsannahme: Wir nehmen an, dass die Werte $P(t, k), \ldots, P(w, k)$ sowie die Werte $P(v, 0), \ldots, P(v, k-1)$ anhand der Rekursionsformel optimal berechnet wurden.

Induktionsschritt: Zu zeigen ist nun, dass die Rekursionsformel auch den Wert $P(v, k)$ optimal berechnet, also dass

$$P(v,k) = \max\{ \sum_{w \in N_v} \{P(w, k_w^1) + c(vw_{k_w^2})| \sum_{w \in N_v} k_w^1 + k_w^2 = k\}$$

ist. Nach Induktionsannahme sind alle Werte $P(w, i)$ mit $i \leq k$ mit $w \in N_v$ optimal berechnet worden, dies bedeutet, dass in den Teilbäumen mit Wurzel w i Bogenkosten erhöht werden dürfen, sodass eine optimale Lösung für den Teilbaum mit Wurzel w vorliegt. Der Teilbaum T_v mit Wurzel v setzt sich nun wie folgt aus den Teilbäumen mit $w \in N_v$ zusammen:
Der Teilbaum $T_v \subseteq T'_S$ besteht aus der Wurzel v, die die Kinder w mit $w \in N_v$ besitzt. Ab jedem einzelnen Knoten w folgt der Teilbaum T_w für $w \in N_v$. v und w sind für jedes $w \in N_v$ über K Bogen verbunden. Angenommen, v hätte nur einen Nachfolger w, dann ist nach Induktionsannahme $\max\{P(w, k_1) + c(vw_{k_2})|k_1 + k_2 = k\}$ der optimale Wert für v. Da v jedoch eine Menge N_v an Nachbarn besitzt, muss dies für alle Nachbarn summiert werden, um das Maximum für den Wurzelbaum T_v zu finden.

Das Dynamische Programm 7 besitzt eine polynomielle Laufzeit:

Es werden maximal $|V(D)|K$ Werte berechnen, wobei für jeden Wert maximal K^{2d} Berechnungen durchgeführt werden. Daraus ergibt sich eine insgesamte Rechenzeit von $\mathcal{O}(|V(D)|K^{2d+1}) = \mathcal{O}(|V(D)|^{2d+2})$, was, da d ein konstanter Wert ist, eine polynomielle Laufzeit beschreibt. ■

Über die Werte $f(v, w, k)$ lässt sich rekursiv, beginnend bei Knoten s und seinen Nachfolgeknoten, ermitteln, wie viele Bogenkosten auf welchen Abschnitten erhöht werden sollen. Da man für das Aufstellen des modifizieren Spannbaums für alle $k \leq K$ auf allen Bogen das Maximalkostenflussproblem mit k variablen Kosten bereits gelöst hat, lässt sich anhand dieser Lösungen nun die optimale Lösung des K-MKF-Problems entnehmen.

Insgesamt ist dies ein exponentielles Verfahren, da es eine exponentielle Anzahl an möglichen Trennungspunkten und auch an Spannbäumen auf diesen Trennungspunkten gibt. Das Verfahren 6 lässt sich, mit einem etwas heuristischeren Hintergrund, zu folgendem Gesamtverfahren modifizieren:

Algorithmus 8 Angepasstes Trennungspunkteverfahren

Input: eine Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ eines K-MKF-Problems
Output: ein kostenmaximaler (b, K)-Fluss f
begin
  while *Noch nicht alle Möglichkeiten betrachtet worden sind* **do**
    Rate eine mögliche Menge der Trennungspunkte;
    Rate einen Spannbaum T auf der Menge der Trennungspunkte;
    Berechne für jeden Abschnitt auf diesem Spannbaum anhand von DP 7 eine Anzahl an Bogen $k \leq K$;
    Entnehme dem angepassten Spannbaum die aktuelle Lösung;
  end
  return *ein kostenmaximaler (b, K)-Fluss anhand eines optimalen Spannbaums und k;*
end

Mit einer geeigneten Wahl der Trennungspunkte oder der Wahl des Spannbaumes müssen nicht alle Möglichkeiten betrachtet werden, was die Laufzeit des Algorithmus 8 durchaus verbessern kann.
Als Trennungspunkte kommen allgemein alle Knoten des Graphen D in Frage. Es kann aber zum Beispiel eine Menge an Trennungspunkten verworfen werden, wenn es im Trennungspunktegraphen einen Knoten $v \in V(G_{\mathcal{T}})$ mit $d^+(v) = 1$ gibt. Solch ein Knoten darf nicht auftreten, da ein Trennungspunkt mindestens zwei Wege trennen soll. Aus gleichen Gründen kann ein Spannbaum verworfen werden, wenn er einen Knoten v mit $d^+(v) = 1$ besitzen sollte. So lässt sich die Laufzeit dieses exponentiellen Verfahrens etwas verbessern. Bogenkapazitäten des Digraphen D können ebenfalls dazu beitragen, dass die Menge der möglichen Trennungspunkte oder die Menge der möglichen Spannbäume ebenfalls weiter eingeschränkt ist.

Dieser Ansatz ist eher kombinatorischer Natur. Das zweite Lösungsverfahren, was wir in diesem Kapitel zur Lösung des K-MKF-Problems sehen werden, greift eine Methode der linearen Optimierung auf.

6.2 Das Column Generation Verfahren

Das Column Generation Verfahren haben wir bereits kurz in Abschnitt 2.6 vorgestellt. Für eine Instanz $(D, s, t_1, \ldots, t_n, \overline{c}, \underline{c}, b, K)$ des K-MKF-Problems gehen wir hier von dem ganzzahligen linearen Modell

Modell 20 Ganzzahliges lineares Programm des K-MKF-Problems

$$\max \sum_{a \in A(D)} \underline{c}(a)y(a) + \sum_{a \in A(D)} (\overline{c}(a) - \underline{c}(a))w(a)$$

$$s.t. \sum_{a \in \delta^+(v)} y(a) - \sum_{a \in \delta^-(v)} y(a) = b(v) \qquad \forall v \in V(D) \qquad (20.1)$$

$$\sum_{a \in A(D)} z(a) \leq K \qquad (20.2)$$

$$z(a) \leq y(a) \qquad \forall a \in A(D) \qquad (20.3)$$

$$w(a) \leq Mz(a) \qquad \forall a \in A(D) \qquad (20.4)$$

$$w(a) \leq y(a) \qquad \forall a \in A(D) \qquad (20.5)$$

$$z(a) \in \{0, 1\} \qquad \forall a \in A(D)$$

$$w(a) \in \mathbb{Z}^+ \qquad \forall a \in A(D)$$

$$y(a) \in \mathbb{Z}^+ \qquad \forall a \in A(D)$$

aus. Standardmäßig lässt sich dieses Modell via eines Branch-and-Bound-Algorithmus mittels LP-Relaxierung lösen. Hier in diesem Abschnitt wollen wir jedoch einen anderen Ansatz zur Lösung dieses Problems wählen, nämlich über die Dantzig-Wolfe Zerlegung mittels Column Generation. Die Nebenbedingungen (20.1) und (20.2) des Modells 20 sind alleinige Nebenbedingungen für die Variablen y und z. In den übrigen Nebenbedingungen treten die Variablen jeweils zusammen auf. Wir betrachten daher für die Dantzig-Wolfe Zerlegung zunächst die Polyeder

$$P^1 := \{y | \sum_{a \in \delta^+(v)} y(a) - \sum_{a \in \delta^-(v)} y(a) = b(v) \text{ für alle } v \in V(D), y(a) \in \mathbb{Z}^+ \text{ für alle } a \in A(D)\}$$

und

$$P^2 := \{z | \sum_{a \in A(D)} z(a) \leq K, z(a) \in \{0, 1\} \text{ für alle } a \in A(D)\}$$

Das Polyeder P^1 beschreibt alle zulässigen Lösungen für einen b-Fluss und das Polyeder P^2 alle Lösungen, bei denen K Einträge von z auf eins gesetzt werden. Mithilfe dieser Polyeder lässt sich das Modell 20 umformulieren.

Modell 21

$$
\begin{aligned}
\max \sum_{a \in A(D)} \underline{c}(a) y(a) + \sum_{a \in A(D)} (\overline{c}(a) - \underline{c}(a)) w(a) & & \\
s.t. z(a) \leq y(a) & \quad \forall a \in A(D) & (21.1)\\
w(a) \leq y(a) & \quad \forall a \in A(D) & (21.2)\\
w(a) \leq M z(a) & \quad \forall a \in A(D) & (21.3)\\
w(a) \in \mathbb{Z}^+ & \quad \forall a \in A(D) & \\
y \in P^1 & & \\
z \in P^2 & &
\end{aligned}
$$

Die verbleibenden Nebenbedingungen (21.1), (21.2) und (21.3) sind die gemeinsame Nebenbedingungen der Variablen y, z, w von Modell 20. Beide Polyeder P^1 und P^2 sind beschränkt, weswegen jeder Punkt $y \in P^1$ als Konvexkombination von Extrempunkten $\{p_1^q\}_{q \in Q^1}$ von P^1 und auch jeder Punkt $z \in P^1$ als Konvexkombination von Extrempunkten $\{p_2^q\}_{q \in Q^2}$ von P^2 geschrieben werden kann. Dabei ist Q^1 die Indexmenge aller Extrempunkte des Polyeders P^1 und Q^2 die entsprechende Indexmenge aller Extrempunkte des Polyeders P^2. Es gilt damit

$$
y = \sum_{i \in Q^1} \lambda_1^i p_1^i \text{ mit } \lambda_1^i \in \mathbb{R}^+ \text{ für alle } i = 1, \ldots, Q^1 \text{ und } \sum_{i \in Q^1} \lambda_1^i = 1
$$

und

$$
z = \sum_{i \in Q^2} \lambda_2^i p_2^i \text{ mit } \lambda_2^i \in \mathbb{R}^+ \text{ für alle } i = 1, \ldots, Q^2 \text{ und } \sum_{i \in Q^2} \lambda_2^i = 1
$$

Diese Darstellung der Variablen y und z setzen wir nun in das Modell 21 ein, um das Masterproblem des Column Generation Verfahrens zu erhalten. In Matrixschreibweise entspricht das Modell 21 der folgenden Formulierung. Dabei ist $\mathcal{I}$ die $|A(D)| \times |A(D)|$-Einheitsmatrix.

Modell 22

$$
\begin{aligned}
\max \quad & \underline{c}^T y + (\overline{c} - \underline{c})^T w & \\
s.t. \quad & z\mathcal{I} \leq yI & (22.1)\\
& w\mathcal{I} \leq y\mathcal{I} & (22.2)\\
& w\mathcal{I} \leq Mz\mathcal{I} & (22.3)\\
& w \in \mathbb{Z}^+_{|A(D)|} & \\
& y \in P^1 & \\
& z \in P^2 &
\end{aligned}
$$

Wir ersetzen nun $\underline{c}_q^i := \underline{c}^T p_q^i$, $\overline{c}_q^i := \overline{c}^T p_q^i$ und $a_q^i := a^T p_q^i$ für $q = 1, \ldots, Q^i$ und $i = 1, 2$. Es ergibt sich somit das folgende Masterproblem.

Modell 23 (Masterproblem)

$$\max \sum_{q \in Q^1} \underline{c}_q^1 \lambda_q^2 + (\overline{c} - \underline{c})^T w$$

$$s.t. \sum_{q \in Q^2} a_q^2 \lambda_q^2 \leq \sum_{q \in Q^1} a_q^1 \lambda_q^1 \tag{23.1}$$

$$w \leq \sum_{q \in Q^1} a_q^1 \lambda_q^1 \tag{23.2}$$

$$w \leq M \sum_{q \in Q^2} a_q^2 \lambda_q^2 \tag{23.3}$$

$$\sum_{q \in Q^1} \lambda_q^1 = 1 \tag{23.4}$$

$$\sum_{q \in Q^2} \lambda_q^2 = 1 \tag{23.5}$$

$$w \in \mathbb{Z}^+_{|A(D)|}$$

$$\lambda_q^i \in \{0, 1\} \qquad \forall q \in Q^i, i = 1, 2$$

Insgesamt wird genau ein Extrempunkt aus Polyeder P^1 und genau ein Extrempunkt aus Polyeder P^2 in einer optimalen Lösung enthalten sein. Da wir jedoch nicht alle möglichen Extrempunkte betrachten wollen, um das Modell 20 zu lösen, wenden wir das Prinzip der Column Generation an. Dabei lösen wir das Masterproblem 21 als beschränktes Masterproblem mit einer Teilmenge an möglichen Lösungen. Die Indexmengen $\overline{Q}^1$ und $\overline{Q}^2$ beschreiben hierbei die gewählten Teilmengen der möglichen Lösungen. Für die Initialisierung ist es wichtig, sinnvoll gewählte Teilmengen zu verwenden.

Modell 24 (Beschränktes Masterproblem)

$$\max \sum_{q \in \overline{Q}^1} \underline{c}_q^1 \lambda_q^1 + (\overline{c} - \underline{c})^T w$$

$$s.t. \sum_{q \in \overline{Q}^2} a_q^2 \lambda_q^2 \leq \sum_{q \in \overline{Q}^1} a_q^1 \lambda_q^1 \qquad [\pi_1] \qquad (24.1)$$

$$1^T w \leq \sum_{q \in \overline{Q}^1} a_q^1 \lambda_q^1 \qquad [\pi_2] \qquad (24.2)$$

$$1^T w \leq M \sum_{q \in \overline{Q}^2} a_q^2 \lambda_q^2 \qquad [\pi_3] \qquad (24.3)$$

$$\sum_{q \in \overline{Q}^1} \lambda_q^1 = 1 \qquad [\nu_1] \qquad (24.4)$$

$$\sum_{q \in \overline{Q}^2} \lambda_q^2 = 1 \qquad [\nu_2] \qquad (24.5)$$

$$w \in \mathbb{Z}^+_{|A(D)|}$$

$$\lambda_q^i \in \{0,1\} \qquad \forall q \in \overline{Q}^i, i = 1,2$$

Bei der Lösung von Modell 24 werden die optimalen primalen Variablen $\overline{\lambda}$ und $\overline{w}$ sowie die optimalen dualen Variablen $\overline{\pi}$ und $\overline{\nu}$ berechnet. Mithilfe der Dualvariablen lassen sich für das Column Generation Verfahren nun die beiden Unterprobleme, die Pricingprobleme, definieren. Dabei entsprechen die Vektoren $c^1 = (c_1^1, \ldots, c^1_{|\overline{Q}^1|})^T$ und $c^2 = (c_1^2, \ldots, c^s_{|\overline{Q}^2|})^T$ den Zielfunktionskosten der Variable y beziehungsweise z im Modell 24. Das bedeutet, dass $c^1 \equiv \underline{c}$ ist und $c^2 \equiv 0$. Außerdem sei $\pi = (\pi_1, \pi_2, \pi_3)^T$ und $\nu = (\nu_1, \nu_2)^T$.

Die Pricingprobleme als Unterprobleme des beschränkten Masterproblems zu den Variablen y und z lauten folgendermaßen:

Modell 25 (Pricingproblem 1 zur Variable y)

$$\max (c^2 - \pi A^1)^T y - \nu_2 \qquad (25.1)$$

$$s.t. \sum_{a \in \delta^+(v)} y(a) - \sum_{a \in \delta^-(v)} y(a) = b(v) \qquad \forall v \in V(D) \qquad (25.2)$$

$$y(a) \in \mathbb{Z}^+ \qquad \forall a \in A(D) \qquad (25.3)$$

mit Matrix $A^1 = \begin{pmatrix} -a_1^1 & \ldots & -a^1_{|\overline{Q}^1|} \\ -a_1^1 & \ldots & -a^1_{|\overline{Q}^1|} \\ 0 & \ldots & 0 \end{pmatrix}$

und

Modell 26 (Pricingproblem 2 zur Variable z)

$$\max(c^1 - \pi A^2)^T z - \nu_1 \tag{26.1}$$

$$s.t. \sum_{a \in A(D)} z(a) \leq K \tag{26.2}$$

$$z(a) \in \{0,1\} \qquad \forall a \in A(D) \tag{26.3}$$

mit Matrix $A^2 = \begin{pmatrix} a_1^2 & \dots & a^2_{|\overline{Q}^2|} \\ 0 & \dots & 0 \\ -Ma_1^2 & \dots & -Ma^2_{|\overline{Q}^2|} \end{pmatrix}$

Es ergibt sich damit das folgende Column Generation Verfahren.

Algorithmus 9 Column Generation

Initialisiere das beschränkte Masterproblem mit einer Teilmenge der möglichen Lösungen;
repeat
  Löse das beschränkte Masterproblem optimal und erhalte die primal optimale Lösung $\lambda^{1*}, \lambda^{2*}$, w und die dual optimalen Lösungen π^* und ν^* ;
  for $i = 1, 2$ **do**
    Löse das Pricingproblem i und erhalte die optimale Lösung c^{*i};
    if $c^{i*} > 0$ **then**
      Sei $j \in Q_i$ mit $c^{i*} = c^i_j$;
      Füge (c^i_j, A^i_j) sowie λ^i_j dem RMP hinzu;
    end
  end
until $c^{*i} \leq 0$ *für* $i = 1, 2$;

Das Pricingproblem zur Variable y nach Modell 25 hat als Flussproblem immer eine ganzzahlige Lösung, da dieser Formulierung eine total unimodulare Matrix zugrunde liegt. Dies bedeutet insbesondere, dass die untere Schranke $\overline{z} \leq z^* \leq \overline{z} + c^{1*} + c^{2*}$ keine Verbesserung gegenüber der Schranke der eigentlichen Formulierung liefert, siehe dazu Lübbecke und Desrosiers [18]. Jedoch existieren für dieses Problem polynomielle Algorithmen. Das bedeutet, dass das ganzzahlige Problem 25 trotz Ganzzahligkeit schnell zu lösen ist, was ein Vorteil ist, da allgemein der Aufwand ein solches ganzzahliges lineares Programm zu lösen, sehr hoch sein kann. Mithilfe von solchen kombinatorischen Algorithmen für das Lösen des Flussproblems und insbesondere der Schnelligkeit, die diese mit sich bringen, lässt sich der Nachteil, dass die Schranke nicht verbessert wird, in diesem Fall aufwiegen.

7 Zusammenfassung und Ausblick

Diese Arbeit behandelte die Problemstellung der Terminvergabe im Krankenhaus, die unter unsicheren klinischen Behandlungspfaden robust sein soll. Die erwünschte Terminvergabe sollte also unter allen möglichen Szenarien eine gute Lösung bestimmen. Dazu haben wir eine Art adaptives robustes Optimierungsmodell aufgestellt. Dieses Optimierungsmodell beinhaltetete zwei Entscheidungsstufen, die Festlegung von Behandlungsreihenfolgen, ohne zu wissen, welche Behandlungen sicher eintreten werden, und die eigentliche Terminvergabe im Max-Szenario-Problem.

Das Max-Szenario-Problem des robusten Modells beinhaltete für eine gegebene Patientenreihenfolge die Suche nach dem schlechtesten Szenario, für dass der beste Zeitplan unter der Zielfunktion, dass die Endzeitpunkte der Behandlungen für alle Patienten in der Summe minimal sind, berechnet werden konnte. Dies konnten wir als eine Art Flussproblem in azyklischen Digraphen interpretieren, in dem - je nach Betrachtung - die Kosten von Bogen oder Knoten erhöht werden durften. Dieses Flussproblem bezeichneten wir als das Maximalkostenflussproblem mit K variablen Kosten. Nachdem wir ein insgesamtes Lösungsverfahren für die robuste Terminvergabe im Krankenhaus unter unsicheren Behandlungspfaden betrachtet haben, untersuchten wir das Maximalkostenflussproblem mit K variablen Kosten in allgemeinen azyklischen Digraphen. Für Digraphen mit einer Senke konnten wir dabei polynomielle Algorithmen finden. Für den allgemeinen Fall von azyklischen Digraphen mit mehreren Senken konnten wir ein Optimalitätskriterium beweisen, aufgrund dessen wir vermuten, dass das Maximalkostenflussproblem mit K variablen Kosten insgesamt $\mathcal{NP}$-vollständig ist. Abschließend gaben wir noch zwei exakte Lösungsverfahren für das Lösen des Maximalkostenflussproblems mit K variablen Kosten an.

Wir haben uns in dieser Arbeit einen kleinen Spezialfall aller Unsicherheiten, die im Krankenhaus auftreten können, ausgewählt und genauer untersucht. Im Krankenhauskontext sind jedoch deutlich mehr Unsicherheiten denkbar, die aus Zeitgründen nicht alle in dieser Masterarbeit behandelt werden konnten. Die Untersuchungen dieser Arbeit beschränkten sich auf unsichere Behandlungspfade, die auch aus graphentheoretischer Sicht Pfaden entsprachen. Für viele medizinische Diagnosen entspricht die graphentheoretische Ansicht der klinischen Behandlungspfade jedoch Bäumen oder Graphen mit Kreisen, wenn eine Behandlung etwa wiederholt werden muss. Für diese Fälle lässt sich die Problemstellung nicht ohne weiteres auf eine Problemstellung des Job Shop Scheduling Problems und auf disjunktive Graphen übertragen, weswegen für eine Lösung unter solchen Behandlungspfaden Anpassungen erforderlich sind oder ein anderer Ansatz gewählt werden muss.

Wir gehen in dieser Arbeit davon aus, dass alle Patienten zu Planungsbeginn und für den gesamten Planungszeitraum zur Verfügung stehen. Außerdem sind alle Patienten bekannt. In realen Planungsproblemen des allgemeinen Krankenhausbereichs ist das eine eher unrealistische Annahme. Zum einen können weitere Patienten wie Notfallpatienten plötzlich während

des Zeitraums in den Krankenhäusern erscheinen, die mit einer höheren Priorität als manche der regulären Patienten behandelt werden müssen. Zum anderen können Patienten nicht während des gesamten Planungszeitraums zur Verfügung stehen. Dafür müssen Releasezeitpunkte und Deadlines in das Modell eingeführt werden. Dies macht die Modellierung allerdings aufwendiger.
Die Annahme, dass keine ungeplanten Patienten auftreten, lässt sich für Krankenhäuser ohne Notaufnahmen, wie private Kliniken oder Rehazentren oder für einzelne Stationen im Krankenhaus realisieren. Ansonsten ist ein Zeitplan nicht mehr optimal einhaltbar, sobald neue Patienten auftreten. Möglichkeiten, einen solchen Terminplan zu retten, wären eingeplante Leerlaufzeiten oder eine nachträgliche Planänderung. Auch kann man einen Terminplan noch ändern, wenn nicht das schlechteste Szenario eingetreten ist, da der Zeitplan so verbessert werden kann. Dies entspricht unserer zweiten Entscheidungsstufe und umfasst den Ansatz, dass ein Plan nach Bekanntwerden eines Szenarios wiederherstellbar ist.

Weitere Unsicherheiten, die wir gar nicht bedacht haben, sind etwa stochastische Behandlungsdauern oder Maschinenausfälle. Allgemein ist auch davon auszugehen, dass Unsicherheiten nicht einzeln auftreten, sondern oftmals in Kombination. Die Modellierung wird somit sehr vielfältig und komplex.

In dieser Arbeit sind wir davon ausgegangen, dass maximal K Behandlungen zusätzlich eintreten können, wobei K ein fest gegebener Wert war. Allgemein ist es wahrscheinlich, dass zum Planungszeitpunkt nicht bekannt ist, wie viele und welche Behandlungen zusätzlich eintreten werden. Im schlimmsten Fall treten alle zusätzlichen Behandlungen auch tatsächlich ein. Auch eine mögliche Variation der Problemstellung, dass nicht K beliebige Behandlungen zusätzlich eintreten werden, von denen jede gleich wahrscheinlich ist, sondern dass für die zusätzlichen möglichen Behandlungen Wahrscheinlichkeiten bekannt sind, mit welcher diese dann eintreten können, ist möglich. Gerade bei typischen Krankheitsverläufen treten sicherlich einige zusätzliche Behandlungen häufiger auf als andere. Auf der Basis von Wahrscheinlichkeiten können ebenso die möglichen zusätzlichen Behandlungen beschrieben werden, falls unklar ist, welche Eintreten können.

Wir haben ein kompaktes Modell 12 der robusten Terminvergabe im Krankenhaus gesehen, was jedoch nicht linear war. In dieser Arbeit befassten wir uns ausschließlich mit linearen Optimierungsmethoden oder graphentheoretischen Problemen. Man kann diese Modelle auch mit quadratischen beziehungsweise konvexen Optimierungsmethoden lösen und so Vergleiche zu den Ergebnissen dieser Arbeit ziehen.

Für das Maximalkostenflussproblem mit K variablen Kosten für allgemeine azyklische Digraphen mit mehreren Senken, losgelöst vom Krankenhauskontext, waren zwei Kostenfunktionen $\overline{c}$ und $\underline{c}$ definiert. Eine weitere interessante Problemstellung wäre die, dass die Kosten nicht zwischen zwei dieser Werte variieren dürfen, sondern dass die Kosten beispielsweise in einem Intervall liegen dürfen oder K Kosten einer einzelnen Kostenfunktion beliebig erhöht werden dürfen.

Die Lösungsverfahren, die in dieser Arbeit in den Kapiteln 4 und 6 vorgestellt wurden, wurden bislang noch nicht in der praktischen Durchführung getestet. Mögliche interessante Vergleiche wären hierbei zum Beispiel, wie sich die beiden Lösungsverfahren für das Maximalkostenflussproblem mit K variablen Kosten über die Trennungspunkte und das Column Generation Verfahren im Vergleich zueinander und zu einer normalen Lösung des Maximalkostenflussproblems mit K variablen Kosten als ganzzahliges lineares Optimierungsproblem verhalten. Auch das Gesamtverfahren zur Ermittlung eines robusten Terminplans im Krankenhaus aus Kapitel 4 wurde bislang noch nicht implementiert und auf Praktikabilität getestet. Insbesondere der Vergleich, wie viel Rechenzeitersparnis die Knotenabschneideregeln erbringen und ob ein anderer Suchbaum zu besseren Ergebnissen führen kann, ist von Interesse. Auch wären Verfahren vorteilhaft, die bessere Schranken für Suchbaumknoten generieren können.

Unklar ist auch noch die genaue Komplexität des Maximalkostenflussproblems mit K variablen Kosten in azyklischen Digraphen mit mehreren Senken. Da wir bislang nur vermuten, dass dies $\mathcal{NP}$-vollständig ist, könnte hierzu der genaue NP-Vollständigkeitsbeweis noch geliefert werden oder ein polynomieller Algorithmus, der das Problem löst. Ebenso können noch weitere, genauere Aussagen für das Maximalkostenflussproblem mit K variablen Kosten getroffen werden.

Abschließend lässt sich festhalten, dass wir in dieser Arbeit eine robuste Terminplanung im Krankenhaus unter unsicheren Behandlungspfaden und dabei ein in der Literatur bisher unbekanntes graphentheoretisches Problem untersucht haben. Im Umfeld dieser Problemstellungen gibt es noch viele offene und noch nicht betrachtete Probleme, die vielfältige Ansätze für zukünftige Arbeiten liefern.

Abbildungsverzeichnis

Algorithmenverzeichnis

Literatur

[1] Aneja, Y. P. ; Aggarwal, V. ; Nair, K. P. K.: Shortest chain subject to side constraints. In: *Networks* 13 (1983), Nr. 2, S. 295–302

[2] Ben-Tal, A. ; Goryashko, A. ; Guslitzer, E. ; Nemirovski, A.: Adjustable robust solutions of uncertain linear programs. In: *Math. Program.* 99 (2004), März, Nr. 2, S. 351–376

[3] Ben-Tal, A. ; Nemirovski, A.: Robust convex optimization. In: *Mathematics of Operations Research* 23 (1998), S. 769–805

[4] Ben-Tal, A. ; Nemirovski, A.: Robust solutions of uncertain linear programs. In: *Operations Research Letters* 25 (1999), S. 1–13

[5] Ben-Tal, Aharon ; Nemirovski, Arkadi: Robust optimization - methodology and applications. In: *Math. Program.* 92 (2002), Nr. 3, S. 453–480

[6] Bertsimas, Dimitris ; Brown, David B. ; Caramanis, Constantine: *Theory and applications of Robust Optimization.* 2007

[7] Bertsimas, Dimitris ; Sim, Melvyn: The Price of Robustness. In: *Operations Research* 52 (2004), Nr. 1, S. 35–53

[8] Brucker, Peter: *Scheduling algorithms.* Springer, 2004. – I–XII, 1–367 S. – ISBN 978-3-540-20524-1

[9] Diestel, Reinhard: *Graphentheorie.* Springer, 2006. – I–XVI, 1–344 S. – ISBN 978-3-540-21391-8

[10] Edmonds, Jack ; Karp, Richard M.: Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. In: *J. ACM* 19 (1972), April, Nr. 2, S. 248–264

[11] Garey, M. R. ; Johnson, David S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979. – ISBN 0-7167-1044-7

[12] Goldberg, Andrew V. ; Tarjan, Robert E.: Finding minimum-cost circulations by canceling negative cycles. In: *J. ACM* 36 (1989), Oktober, Nr. 4, S. 873–886

[13] Jongen, Hubertus T. ; Meer, Klaus ; Triesch, Eberhard: *Optimization theory.* Kluwer, 2004

[14] Kapoor, Sanjiv ; Ramesh, H.: An Algorithm for Enumerating All Spanning Trees of a Directed Graph. In: *Algorithmica* 27 (2000), S. 120–130

[15] Korte, Bernhard ; Vygen, Jens: *Combinatorial Optimization: Theory and Algorithms.* 4th. Springer Publishing Company, Incorporated, 2007. – ISBN 3540718435, 9783540718437

[16] Liebchen, Christian ; Lübbecke, Marco ; Möhring, Rolf H. ; Stiller, Sebastian: *Recoverable Robustness.* 2007

[17] Liu, Nan ; Ziya, Serhan ; Kulkarni, Vidyadhar G.: Dynamic Scheduling of Outpatient Appointments Under Patient No-Shows and Cancellations. In: *Manufacturing & Service Operations Management* 12 (2010), Nr. 2, S. 347–364

[18] Lübbecke, M.E. ; Desrosiers, J.: Selected Topics in Column Generation. In: *Oper. Res.* 53 (2005), Nr. 6, S. 1007–1023

[19] Paulussen, Torsten O. ; Jennings, Nicholas R. ; Decker, Keith S. ; Heinzl, Armin: Distributed Patient Scheduling in Hospitals. In: Gottlob, Georg (Hrsg.) ; Walsh, Toby (Hrsg.): *IJCAI*, Morgan Kaufmann, 2003, S. 1224–1232

[20] Paulussen, Torsten O. ; Zöller, Anja ; Heinzl, Armin ; Braubach, Lars ; Pokahr, Alexander ; Lamersdorf, Winfried: Patient scheduling under uncertainty. In: Haddad, Hisham (Hrsg.) ; Omicini, Andrea (Hrsg.) ; Wainwright, Roger L. (Hrsg.) ; Liebrock, Lorie M. (Hrsg.): *SAC*, ACM, 2004, S. 309–310

[21] Schlüchtermann, J.: *Patientensteuerung.* Verlag Josef Eul, Bergisch-Gladbach ,Germany, 1990

[22] Schrijver, Alexander: *Combinatorial Optimization: Polyhedra and Efficiency.* Heidelberg : Springer, 2003

[23] Soyster, A. L.: Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming. In: *Operations Research* 21 (1973), Nr. 5, S. 1154–1157

[24] Villeneuve, D. ; Desrosiers, J. ; Lübbecke, M.E. ; Soumis, F.: On Compact Formulations for Integer Programs Solved by Column Generation. In: *Ann. Oper. Res* 139 (2005), Nr. 1, S. 375–388

[25] Ziegelmann, Mark: *Constrained Shortest Paths and Related Problems*, Universität des Saarlandes, Doctoral dissertation, 2001

Stichwortverzeichnis

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Aachen, 30.September 2013 .....
Luisa Eickmeyer