Das Resource-Constrained-Shortest-Path-Problem und seine Anwendung in der ÖPNV-Dienstplanung

Diplomarbeit

von

Thomas Schlechte

Technische Universität Berlin

Studiengang: Wirtschaftsmathematik

Berlin, den 5. November 2003

Erstgutachter:	**Professor Dr. M. Grötschel**
Zweitgutachter:	**Professor Dr. R. Möhring**
Betreuer:	**Dr. R. Borndörfer**

Hiermit versichere ich die selbständige und eigenhändige Anfertigung an Eides statt.

Thomas Schlechte

Berlin, den 5. November 2003

Danksagung

Mein herzlicher Dank gilt den vielen Menschen, die zum Enstehen dieser Arbeit beigetragen haben.

In erster Linie möchte ich Dr. Ralf Borndörfer, Steffen Weider und Dr. Andreas Löbel für die Zusammenarbeit und die Unterstützung danken. Ohne sie und ohne die geistige und technische Infrastruktur des Konrad-Zuse-Zentrum für Informationstechnik Berlin wäre diese Diplomarbeit nicht möglich gewesen.

Mein persönlicher Dank gilt meiner Familie, meinen Freunden und all den Weggefährten, die mir in dieser Zeit den nötigen Rückhalt gaben.

Danke

Thomas Schlechte

Berlin, den 5. November 2003

Dankbarkeit durch geschriebene Worte auszudrücken wird dem eigentlichen Gefühl selten gerecht, dennoch wird diese heuristisch ermittelte, zulässige Lösung auch in dieser Diplomarbeit verwendet.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Verkehrsplanung und Optimierung

Die Verkehrssysteme des 21. Jahrhunderts stehen heute vor zahlreichen Herausforderungen, aber auch vor ebenso vielen Chancen. Globalisierung im Allgemeinen, ein rasantes Zusammenwachsen Europas im Speziellen, die Liberalisierung von einst monopolistischen Märkten und eine immens steigende und vor Jahrzehnten noch undenkbare Mobilität der Bevölkerung gilt es zu bewältigen. Verschiedenste Wissenschaften versuchen diesen Anforderungen gerecht zu werden. Fahrzeugtechnologie, Verkehrslogistik, Betriebswirtschaft, aber auch Informatik und Mathematik spielen dabei die Hauptrollen. Der Einsatz von IT-Systemen, die die dabei auftretenden Massen an Daten und Informationen verwalten und analysieren, ist ebenso unumgänglich wie der Einsatz von Optimierungssoftware für die komplexen Verkehrsplanungsprobleme. Das Thema dieser Arbeit, die Dienstplanung, ist dabei ein einzelner Schritt der operativen Planung im öffentlichen Personennahverkehr. Eine thematische Einstimmung und die Erklärung aller relevanten Begriffe der Dienstplanung im ÖPNV werden im ersten Kapitel gegeben. Die Struktur der gesamten Diplomarbeit ist in Abbildung 1.1 festgehalten und verdeutlicht die scheinbar einfache Vorgehensweise der Wirtschaftsmathematik. Ein ökonomisches, reales und komplexes Problem wird sinnvoll mathematisch modelliert und somit auf ein abstraktes und allgemeines Niveau gehoben. Dadurch wird der Einsatz mathematischer Methoden ermöglicht, deren Zusammenhang zum Ausgangsproblem keinesfalls offensichtlich war. Auf diese Weise ist man letztendlich in der Lage, mit Hilfe sich stets weiterentwickeltender Informationstechnik, Rechnerleistung, Algorithmen und Implementationen Probleme mit Millionen Variablen zu bearbeiten und nahezu optimale Lösungen zu bestimmen.

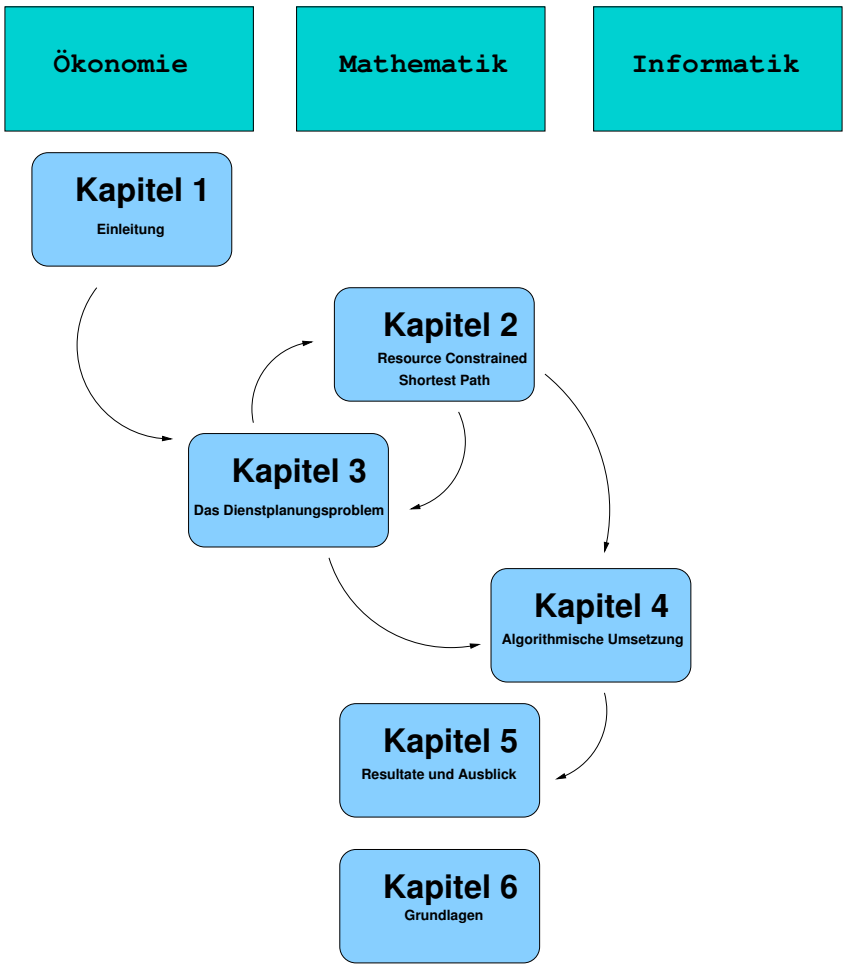

Abbildung 1.1: Gliederung der Arbeit.

1.2 Problemstellung im öffentlichen Personennahverkehr

Die Aufgabenstellung im ÖPNV ist es, aus einer Menge elementarer Arbeitseinheiten, sog. *Dienstelemente*, eine Menge von Tagesdiensten für einzelne Fahrer zu bilden. Diese Tagesdienste sind, ganz allgemein betrachtet, zulässige Abfolgen von Dienstelementen. Eine Menge von Diensten, die alle auszuführenden Arbeitseinheiten abdeckt, soll fortan mit *Dienstplan* bezeichnet werden. Diese Dienstpläne sollen bzgl. gewählter Zielkriterien optimiert werden, denkbar sind hier die Personalkosten, Anzahl der Dienste, aber auch soziale und betriebliche Vorgaben bzgl. der Arbeitszeit. Zu beachten ist bei der Planung, dass gesetzliche, tarifliche, technische, betriebliche und andere Anforderungen existieren, die sowohl für einzelne Dienste als auch für den Dienstplan als Ganzes erfüllt sein müssen. Bei den benötigten Definitionen und Begriffen halte ich mich vor allem an die Arbeit [BLSV98] zur Dienst- und [LS95] zur Umlaufplanung.

1.3 Die vier Ebenen der Dienstplanung

Die Abbildung 1.2 beschreibt das Problem der Dienstplanung global. Zum einen verdeutlicht sie die Position der Dienstplanung in der gesamten sequentiellen Planungsabfolge, und zum anderen gibt sie Aufschluss über eine mögliche atomare Sichtweise von Dienstplänen und Diensten, der wir uns im folgenden Abschnitt von oben nach unten nähern wollen.

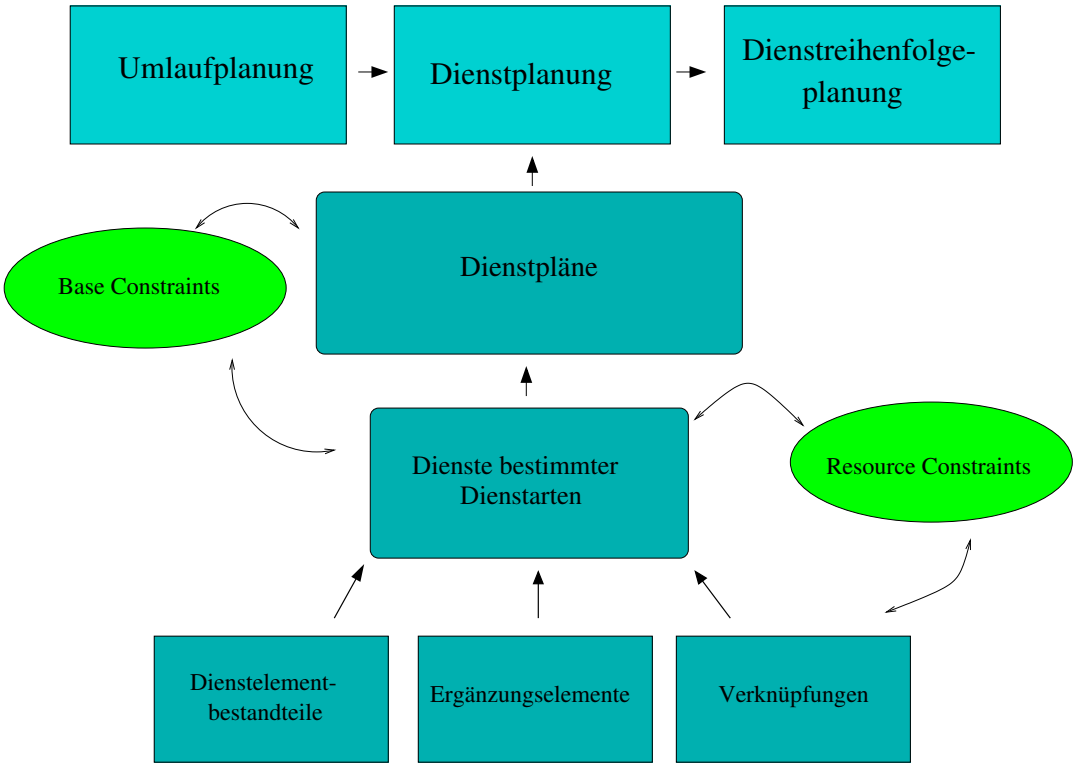

Abbildung 1.2: Die vier Ebenen der Dienstplanung im ÖPNV.

Der Dienstplan

Eine Menge von Diensten, in der jedes Dienstelement genau einmal von einem Dienst abgedeckt wird, heißt Dienstplan. In gewissen Szenarien ist aber auch eine mehrfache Abdeckung erlaubt. Es besteht auch die Möglichkeit, Dienstelemente als optional zu betrachten, d.h. solche Elemente können, müssen jedoch nicht abgedeckt werden. Der Umgang mit solchen spezifischen Vorgaben wird bei der Problemmodellierung genauer erläutert. Hier soll vorerst ein Überblick über die Terminologie im ÖPNV gegeben werden.
Der Dienstplan selbst kann aber auch Bedingungen unterliegen. So ist zum Beispiel ein ausgewogener Dienstmix, d.h. ein vorgegebenes Verhältnis von Dienstarten oder Diensttypen zu berücksichtigen. Diese Sachverhalte werden meist unterteilt in weiche und harte Nebenbedingungen. Harte Nebenbedinungen werden direkt als sog. *Base Constraints* formuliert, während

Abweichungen von weichen Nebenbedingungen in geeigneter Art und Weise in der Zielfunktion bestraft werden.

Spezifikation der Dienste durch Dienstarten

Durch eine konkrete Festlegung der unten erläuterten Parameter wird eine *Dienstart* definiert. In der Dienstplanung werden nur Dienste betrachtet, die genau einer spezifizierten Dienstart entsprechen.

- **Pausenregelungen** legen gemäß der gesetzlichen Lenkzeitverordnung fest, wann und wie lange eine Pause stattfinden muss, bzw. in welchem Umfang eine Unterbrechung überhaupt als Pause anrechenbar ist. Zu erwähnen sind hierbei vor allem Blockpausen- und Quotientenregelungen. Eine vollständige und umfassende Definition jenes Pausenregelwerkes folgt an geeigneter Stelle in Kapitel 4.
- Unter dem Begriff **Diensttypen** wird die temporale Struktur des Dienstes spezifiziert, d.h. ob es sich um einen Früh-, Mittel-, Spät oder Nachtdienst und ob es sich um einen zeitlich zusammenhängenden oder um einen sog. *geteilten Dienst* handelt.
- **Dienstuntereinheiten** sind Dienstteile und Dienststücke.
  Ein Dienststück entspricht dem maximalen Teil eines Umlaufes, der von einem Fahrer unmittelbar aufeinanderfolgend gefahren wird. Diese Fahrzeugumläufe sind Ergebnis der Umlaufplanung und stehen bereits fest. Vereinfacht dargestellt beginnt ein Dienststück mit dem Betreten des Fahrzeugs und endet bei dessen Verlassen.

  Durch die Existenz von Verkehrsspitzen gibt es geteilte Dienste, bei denen eine oder mehrere größere Unterbrechungen, die nicht zum Dienst zählen und die insbesondere nicht als Pause anrechenbar sind, auftreten. Diese Dienstteile bestehen aus 'mindestens' einem Dienststück.
- Weitere **Attribute** sind Schichtlänge, die Anzahl und Dauer der Dienstteile bzw. -stücke, Lenk-, Arbeits- und Pausenzeit. Für all diese Parameter werden in der Praxis sowohl minimale, maximale als auch durchschnittliche oder fest angestrebte Werte verwendet. Wenn es sich hier um lineare Zusammenhänge handelt, wird dies durch sogenannte *Resource Constraints* modelliert.

Atome der Dienstplanung

- Die bereits erwähnten **Dienstelemente** sind der Input der Dienstplanung und werden im Allgemeinen als Ergebnis der Umlaufplanung durch Zerlegen der Fahrzeugumläufe an sog. Ablösepunkten gewonnen. Wie grob oder fein hier die einzelnen Umläufe zerlegt werden, ist ein erster Freiheitsgrad der Planung. Dienstelemente sind also Arbeitseinheiten, die von einem Fahrer zwingend hintereinander (am Stück) erledigt werden müssen. Da Dienstelemente somit heterogene Sequenzen sind, ist eine weitere Unterteilung in **Dienstelementbestandteile** zur Vereinfachung der Implementation sinnvoll. Dienstelementbestandteile haben somit genau einen zusammenhängenden Lenk-, Arbeits- und Pausenzeitanteil, sowie eine feste Start- und Endzeit.

- **Ergänzungselemente** hingegen sind Zusatzarbeiten, die zu Beginn oder am Ende eines Dienststückes bzw. -teiles anfallen. Diese sind somit abhängig von den einzelnen Diensten (und unterliegen einer genauer zu definierenden Positionslogik). Hier liegt auch der entscheidende Unterschied zu den Dienstelementen. Während diese alle von mindestens einem zulässigen Dienst erfüllt werden müssen, ergibt sich aus den generierten Diensten des gewählten Dienstplanes, welche Ergänzungselemente durchzuführen sind.

- **Verknüpfungen** modellieren die lokale Kombinierbarkeit dieser Elemente zu Tagesdiensten. Hierbei ist zu beachten, dass nicht alle Sequenzen von Elementen zulässig sind, sei es aus betrieblichen, lokalen oder zeitlichen Gründen. Elemente die nach dieser Betrachtung hintereinander auftreten könnten, werden durch eine Verknüpfung miteinander verbunden, wobei diese wiederum einen anrechenbaren Pausenanteil aufweisen kann. Nur eine angemessene Auswahl dieser Verknüpfungsregeln kann zu einerseits handhabbaren und andererseits optimierbaren Probleminstanzen führen. Um das Optimierungspotential ausschöpfen zu können, arbeiten wir jedoch mit so vielen Freiheitsgraden wie möglich.

Betrachtet man nun die Abbildung 1.2 aus Sicht der Atome, so sind Dienste zulässige Sequenzen von Dienstelementbestandteilen, Ergänzungselementen und Verknüpfungen, d.h. zulässige Abfolgen von Fahrgastfahrten, Leerfahrten, Fahrten vom bzw. zum Depot, sowie anderer relevanter Tätigkeiten des Fahrpersonals im ÖPNV.
Eine Menge von Diensten, die alle zu erfüllenden Aufgaben abarbeitet und alle weiteren Base Constraints erfüllt, entspricht somit einem zulässigen Dienstplan. Unter all diesen zulässigen Dienstplänen gilt es denjenigen auszuwählen, der für die gewünschte Zielfunktion den Optimalwert annimmt.
Mathematisch werden Dienste graphentheoretisch als Pfade in einem Digraphen modelliert, wobei Dienstelementbestandteile und Ergänzungselemente als Knoten und Verknüpfungen als Bögen zu interpretieren sind.
In Kapitel 2 werden allgemein ein Optimierungsproblem in Digraphen vorgestellt und verschiedenste Lösungsalgorithmen diskutiert. Erst in Kapitel 3 wird die Verbindung zwischen dem in dieser Einleitung beschriebenen Diensplanungsproblem und dem abstrakten Problem des zweiten Kapitels deutlich.

Integration in die gesamte operative Planung

Hier ist zu erwähnen, dass die beste Optimierung nur für sinnvolle Instanzen von Nutzen ist, denn nur ein geeignetes Zusammenspiel der vorhergehenden und nachfolgenden Planungsschritte kann in der Praxis zu guten Gesamtlösungen führen. Dass dieser sequentielle Ablauf der Planung Potentiale unausgeschöpft lässt, zeigen aktuelle Bemühungen, integrierte Dienst- und Umlaufplanungsmodelle und -algorithmen zu entwickeln, und die Ergebnisse aus den Veröffentlichungen [FHW00] und [BLW02].

Kapitel 2

Das Resource Constrained Shortest Path Problem

Während das erste Kapitel die Motivation und Problematik in der Dienstplanung dargelegt hat, soll nun eine allgemeine Darstellung der Lösungsideen und -algorithmen für das Resource Constrained Shortest Path Problem (RCSP) folgen. Zwar existieren dazu bereits umfangreiche und weitgreifende Übersichtsarbeiten von I. Dumitrescu [D02] und M. Ziegelmann [Z01], dennoch ist es an dieser Stelle notwendig, die wichtigsten Aussagen und Herangehensweisen zu erläutern, um den späteren Argumentationen bei der Lösung der gegebenen Instanzen folgen zu können.

2.1 RCSP Modell

Das kürzeste Wege Problem mit weiteren Nebenbedingungen (resource constrained shortest path problem) ist die Frage nach einem Pfad P in einen Digraphen $D = (V, A)$, der linearen Ressourcenbedingungen genügt und kostenminimal ist. Sei dazu die Knotenmenge $V = \{0, 1, 2 \dots n\}$, die Bogenmenge $A \subseteq V \times V$ mit $|A| = m$ und der maximal zulässige Verbrauch λ^i der Ressourcen $i = 1, 2 \dots k$ gegeben. Jedem Bogen $a \in A$ werden nun ein Kostenwert c_a und Ressourcenwerte r_a^i zugeordnet, die bei einer entsprechenden Nutzung verbraucht werden. Des Weiteren seien die Knoten s und t als Quelle und Senke des gesuchten Pfades markiert. Dann lässt sich das RCSP folgendermaßen formulieren:

- Knoten-Bogen-Inzidenzmatrix $N \quad \in \{-1, 0, 1\}^{n \times m}$
- Ressourcenmatrix $R = (r_a^i) \quad \in \mathbb{R}^{k \times m}$
- Kostenvektor $c \quad \in \mathbb{R}^m$
- Kapazitätsvektor $\lambda \quad \in \mathbb{R}^k$

$$
\begin{aligned}
(RCSP) \quad \min \quad & c^T x && && && (3.1)\\
\text{s.t.} \quad & Nx && = && e_t - e_s && (3.2)\\
& Rx && \leq && \lambda && (3.3)\\
& \sum_{a \in (S \times S) \cap A} x_a && \leq && |S| - 1 \quad \forall\, S \subseteq V && (3.4)\\
& x && \in && \{0,1\}^{|A|} && (3.5)
\end{aligned}
$$

Hierbei gibt x an, welche Bögen im Pfad enthalten sind. Die Bedingungen (3.2) stellen zum einen sicher, dass es sich im gegebenen Digraphen um einen Pfad von s nach t handelt, d.h. genau ein Bogen verlässt den Knoten s und genau ein Bogen erreicht den Knoten t. Zum anderen wird sicher gestellt, dass für jeden anderen Knoten die Anzahl der Bögen, die diesen erreichen, genau der Anzahl der Bögen entspricht, die diesen verlassen.
Die Bedingungen (3.4), bekannt als *Subtour Elimination Constraints*, gewährleisten, dass keine geschlossenen Kreise zulässig sind. Dadurch ist insgesamt die Menge der zulässigen Lösungen auf die elementaren Pfade beschränkt, d.h kein Knoten oder Bogen wird mehrmals besucht.
Dass der Pfad alle Ressourcenbeschränkungen erfüllt, stellen abschließend die Bedingungen (3.3) sicher. Aus Konventionsgründen und um Vergleiche mit den an späterer Stelle diskutierten Arbeiten möglich zu machen, sind die Ressourcenbedingungen hier ausschließlich als $\leq$ Bedingung formuliert. Wegen der bekannten Transformationsmöglichkeiten zwischen Gleichungen und Ungleichungen aus der linearen Optimierung stellt dies aber keine Einschränkung dar. Gesetzt dem Fall, dass man durchaus Pfade zulassen will, die nicht elementar sind, wie der in Abbildung 2.1 dargestellte, kann man anstatt der Bedingungen (3.4) andere Bedingungen modellieren, um die in Abbildung 2.2 dargestellten Pfad-Kreis-Lösungen dennoch abzuschneiden.
Man beachte jedoch, dass in azyklischen Digraphen trivialerweise keine nicht elementaren Pfade existieren, so dass in diesem Fall die Bedingungen (3.4) redundant sind. Die Formulierung des Problemes als ein Integer Program (IP) wirft Fragen hinsichtlich der Komplexität und Lösungsmöglichkeit auf. Diese werden im folgenden Abschnitt thematisiert.

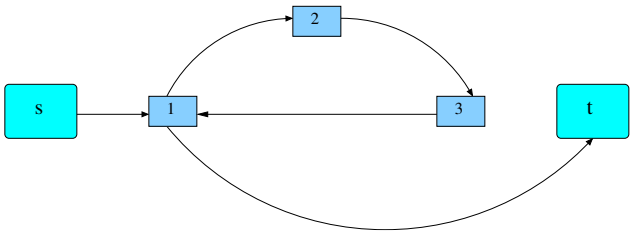

Abbildung 2.1: Ein nicht elementarer (s,t)-Pfad.

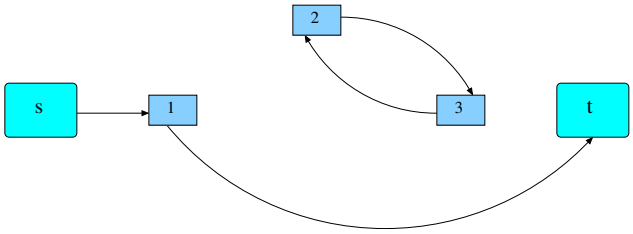

Abbildung 2.2: Eine entartete Pfad-Kreis-Lösung.

2.2 Komplexität

Das einfache Shortest Path Problem ist längst sowohl detailliert in der Theorie untersucht, als auch praktisch effizient implementiert worden. Die wichtigsten Aussagen dazu sollen hier dennoch Erwähnung finden, da sie trivialerweise auch für das RCSP von Bedeutung sind. Für das kürzeste Wege Problem in einem Digraphen mit positiven Gewichten existieren polynomiale Markierungs- bzw. Labelingalgorithmen, die vor allem auf die Arbeit [D59] von Dijkstra zurückgehen.
Für beliebige Gewichte ist jedoch das kürzeste Wege Problem in einem Digraphen äquivalent zum Problem, einen längsten Weg mit beliebigen Gewichten in einem Digraph zu bestimmen. Letzteres Problem ist aber $\mathcal{NP}$-vollständig, da eine einfache Transformation auf das Entscheidungsproblem, ob ein Digraph einen gerichteten Hamiltonschen Weg enthält, existiert. Somit ist das einfache Shortest Path Problem mit beliebigen Gewichten $\mathcal{NP}$-schwer. Andererseits existieren, falls der Digraph keinen gerichteten negativ gewichteten Kreis enthält, auch für beliebige Gewichte polynomiale Lösungsalgorithmen. Alle hier nicht genauer ausgeführten Komplexitätsaussagen und Algorithmen zum Shortest Path Problem lassen sich in den Werken [AMO93], [M84] und [S03] nachlesen.
Während also für das Bestimmen kürzester Wege in einem Digraphen mit positiven Bogengewichten längst polynomiale und somit effiziente Algorithmen existieren, wird das Problem schon durch Hinzunehmen einer Ressourcenbedingung $\mathcal{NP}$ vollständig. Dies belegen die Arbeiten [GJ79] und [HZ80]. Während Garey und Johnson eine Transformation des Partion Problem zum RCSP Problem konstruierten, transformierten Handler und Zang das Knapsackproblem zu einem RCSP Problem. Die Verbindung zum Knapsackproblem zu erläutern, erscheint mir dabei am sinnvollsten, da man zum einen ein besseres Verständnis für das Problem erhält und zum anderen fällt es leichter, den bestehenden algorithmischen Lösungsideen für das RCSP zu folgen. Daher soll diese Transformation hier näher betrachtet werden. Dazu sei das Knapsackproblem (KP) wie folgt definiert:
Gegeben sei eine Menge von n-Elementen, wobei jedes Element einen gewissen Wert und ein spezielles Gewicht hat. Das Ziel ist es, eine Teilmenge dieser Menge zu bestimmen, die ein gegebenes Gewichtslimit λ nicht übersteigt und deren Gesamtwert maximal ist. Das Problem, einen Rucksack so voll zu packen, dass das Wichtigste hineinpasst, scheint hier die namensgebende reale Problemstellung gewesen zu sein. Somit lässt sich das KP mit folgenden Daten definieren:

- Wertvektor v $\in \mathbb{N}_+^n$
- Gewichtsvektor w $\in \mathbb{N}_+^n$
- Gewichtslimit λ $\in \mathbb{N}_+$

$$(KP) \quad \max v^T x \tag{2.1}$$
$$w^T x \leq \lambda \tag{2.2}$$
$$x \in \{0,1\}^n \tag{2.3}$$

Es wird also über die Inzidenzvektoren aller Teilmengen der n-elementigen Grundmenge optimiert. Die vorausgesetzte Positivität von v und w stellt keine allzu grosse Einschränkung des Problems darstellt. Während negativ bewertete Elemente mit positivem Gewicht trivialerweise nicht in der optimalen Teilmenge enthalten sind, können negativ gewichtete Elemente die Gewichtsrestriktion nie verletzen und somit bedenkenlos zur Teilmenge hinzugefügt werden, falls diese den Zielfunktionswert erhöhen. Problematisch sind nur Elemente, deren Gewicht und Wert negativ sind. Andere Preprocessing Überlegungen könnten zum Beispiel das Weglassen von Elementen $i \in \{1, 2 \ldots, n\}$ mit $a_i > \lambda$ sein.
Das KP ist äquivalent zu einem RCSP, welches dazu wie folgt konstruiert wird:
Sei $D = (V, A)$ ein Digraph mit einer $(n+1)$-elementigen Knotenmenge V und folgender Bogenmenge:

$$A = \{(i, i+1)^{(k)} \text{ mit } i \in \{1, 2 \ldots, n\} \subset V \text{ und } k \in \{1, 2\}\}$$

Die Kosten $c_{i,i+1}^{(k)}$ und der Ressourcenverbrauch $r_{i,i+1}^{(k)}$ der Bögen $(i, i+1)^{(k)}$ wird nun wie folgt festgelegt:

Sei $M = \max\limits_{i \in \{1,2\ldots,n\}} \{v_i\}$, dann setze

$$r_{i,i+1}^{(k)} = \begin{cases} w_i & , k = 1 \\ 0 & , k = 2 \end{cases}$$

und

$$c_{i,i+1}^{(k)} = \begin{cases} M - v_i & , k = 1 \\ M & , k = 2. \end{cases}$$

Das Finden eines kürzesten Weges bzgl. c, der die Ressourcenbedingung bzgl. r und λ erfüllt, liefert eine Teilmenge, die die Knapsackungleichung nicht verletzt und maximal bzgl. v ist. Dies funktioniert wie folgt:
Jeder $(1, n+1)$-Pfad enthält entweder den Bogen $(i, i+1)^{(1)}$ oder $(i, i+1)^{(2)}$. Führt der Pfad nun über den Bogen $(i, i+1)^{(2)}$ so bedeutet dies, dass das Element i nicht in der Teilmenge enthalten ist, andernfalls liegt der Bogen $(i, i+1)^{(1)}$ auf dem Pfad und die zum Pfad korrespondierende Teilmenge enthält das Element i. Offenbar ist die so konstruierte Teilmenge ein Optimum des KP, denn gäbe es eine Teilmenge, die die Knapsackungleichung erfüllt und einen echt größeren Zielfunktionswert aufweist, als die von einem ressourcenbeschränkten kürzesten Pfad konstruierte, dann würde diese wiederum mit einem zulässigen Pfad korrespondieren (mit

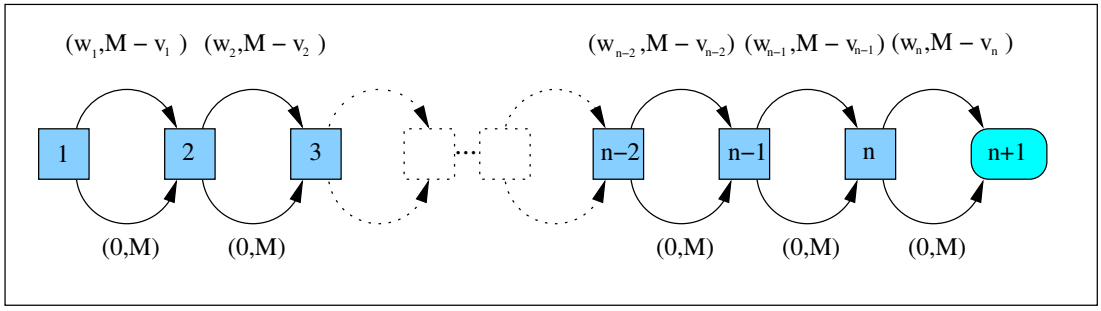

Abbildung 2.3: RCSP-Knapsack Transformationsnetzwerk.

der eben erwähnten Transformation), der die Ressourcenbedingung erfüllt und echt kürzer ist als der kürzeste, was direkt zu einem Widerspruch führt.
Insgesamt führt dies zu den zwei folgenden komplexitätstheoretischen Erkenntnissen:
Zum einen liegt RCSP in $\mathcal{NP}$, da der Pfad selbst als Zertifikat dient und es möglich ist, in polynomialer Zeit zu entscheiden, ob er die Ressourcenbeschränkung erfüllt. (Man beachte, dass es sich hier um das vom RCSP induzierte Entscheidungsproblem, d.h. 'Gibt es einen Weg der die Ressourcenbeschränkung erfüllt und nicht länger ist als ein festes $\overline{c}$?' handelt). Da nun bekanntermaßen das zum Optimierungsproblem KP gehörende Entscheidungsproblem $\mathcal{NP}$-vollständig ist und die oben formulierte Transformation polynomial in der Kodierungslänge des Inputs implementiert werden kann, ist gezeigt, dass das RCSP $\mathcal{NP}$-schwer ist.
Dies führt zur zweiten, noch stärkeren Aussage, dass sich jedes Entscheidungsproblem $\Pi \in \mathcal{NP}$ polynomial in der Kodierungslänge von Π auf das zum RCSP zugehörige Entscheidungsproblem transformieren lässt.

2.3 Untere Schranken für das RCSP

2.3.1 Lösen der LP Relaxierung

Um zu zeigen, dass man die LP Relaxierung des RCSP schnell und effizient lösen kann, scheint folgende alternative Modellierung des RCSP äußerst hilfreich. Sei für jeden (elementaren) Pfad $p \in \mathcal{P}$ $(x)_p$ eine 0/1-Variable. Man beachte, dass in der Regel in einem beliebigen Digraphen exponentiell viele (elementare) Pfade existieren. Des Weiteren gilt für die Kosten c_p und den Ressourcenverbrauch $r_{i,p}$ eines Pfades:

$$\begin{aligned} c_p &= \sum_{e \in p} c_e \quad \forall p \in \mathcal{P} \\ r_{i,p} &= \sum_{e \in p} r_e^i \quad \forall p \in \mathcal{P}, \forall i = 1, 2, \ldots k \end{aligned}$$

Dies ist keine Definition im eigentlichen Sinne, vielmehr die intuitive Transformation der Bogenparameter auf Pfadparameter, so dass die zulässigen Lösungen in all ihren Eigenschaften

unverändert bleiben. So lässt sich das RCSP' in Pfadvariablen $x \in \{0,1\}^{|P|}$ mit Kostenvektor $c \in \mathbb{R}^{|P|}$ und Ressourcenverbrauchsmatrix $R \in \mathbb{R}^{k \times |P|}$ wie folgt formulieren:

$$\begin{aligned} \text{(RCSP')} \quad \min \quad & c^T x && \text{(3.1a)} \\ \text{s.t.} \quad & \mathbb{1}^T x = 1 && \text{(3.2a)} \\ & Rx \leq \lambda && \text{(3.3a)} \\ & x \in \{0,1\}^{|\mathcal{P}|} && \text{(3.4a)} \end{aligned}$$

Relaxiert man nun die Ganzzahligkeitsbedingung (3.4a) und dualisiert das so entstandene LP, so erhält man mit $u \in \mathbb{R}$ und $\pi \in \mathbb{R}^k$ das folgende lineare Programm $DP_{RCSP'}$:

$$\begin{aligned} \text{(DP}_{\text{RCSP'}}\text{)} \quad \max \quad & u + \lambda^T \pi && \text{(3.1b)} \\ \text{s.t.} \quad & u\mathbb{1} + R^T \pi \leq c && \text{(3.2b)} \\ & \pi \leq 0 && \text{(3.3b)} \end{aligned}$$

Somit haben wir die LP Relaxierung des RCSP in ein lineares Programm mit $k+1$ Variablen, aber exponentiell vielen (für jeden Pfad $p \in \mathcal{P}$ eine) Ungleichungen überführt. Es folgt aus den Erkenntnissen über die Zusammenhänge von Optimieren und Separieren, die ausführlich in [GLS88] nachzulesen sind, dass es ausreicht, das Separierungsproblem für das $\text{DP}_{\text{RCSP'}}$ in polynomialer Zeit zu lösen, d.h. folgende Fragestellung für ein gegebenes $\overline{u}$ und $\overline{\pi}$ zu beantworten:

$$\begin{gathered} \exists\, q \in \mathcal{P} \;:\; \overline{u} + R_q^T \overline{\pi} > c_q \\ \Leftrightarrow \\ \exists\, q \in \mathcal{P} \;:\; c_q - R_q^T \overline{\pi} < \overline{u} \\ \Leftrightarrow \\ \exists\, q \in \mathcal{P} \;:\; c_q - \sum_{i=1}^{k} r_{i,q} \overline{\pi}_i < \overline{u} \\ \Leftrightarrow \\ \exists\, q \in \mathcal{P} \;:\; \sum_{e \in q} \overline{c}_e < \overline{u} \end{gathered}$$

Die Existenz eines solchen Pfades lässt sich also leicht überprüfen, denn dieses Problem entspricht der Frage, ob ein Pfad q im Digraphen D existiert, der bzgl. der transformierten

Bogengewichte $\overline{c}_e = c_e - R_e^T \pi$ nicht länger als $\overline{u}$ ist.
Unter der Annahme, dass die Kosten und die Ressourcenverbrauchswerte nicht negativ sind, sind auch die transformierten Bogengewichte nicht negativ. Dadurch lässt sich dies in polynomialer Zeit effizient mit einem Dijkstra-Algorithmus testen, womit auf theoretischer Ebene gezeigt wurde, dass die LP Relaxierung des RCSP in polynomialer Zeit zu lösen ist. Man beachte, dass dieses Resultat auch für zykelfreie Graphen mit negativen Bogengewichten Bestand hat.

2.3.2 Lösen der Lagrange Relaxierung

Eine mindestens genauso gute untere Schranke wie die der LP Relaxierung lässt sich durch das Lösen der Lagrange Relaxierung gewinnen, d.h. genauer durch das Finden der optimalen Lagrange Multiplikatoren. Diesen allgemein gültigen Zusammenhang zwischen LP und Lagrange Relaxierung kann man in [NW88] nachlesen. Dieser Ansatz ist vor allem durch die geeignete Struktur des RCSP sinnvoll. Die Resource Constraints, die das Problem eigentlich erst $\mathcal{NP}$-schwer 'werden lassen', werden mit Lagrange Multiplikatoren α in die Zielfunktion als 'Strafen' eingebaut.
Das so entstehende kürzeste Wege Problem mit transformierten Kosten, die ein Überschreiten der Ressourcen bestrafen, kann wiederum, wie bereits mehrfach erwähnt, unter speziellen Annahmen mit einem Dijkstra-Algorithmus in polynomialer Zeit gelöst werden. Insgesamt liefert der Optimalwert $z_{LR}(\alpha)$ mit $\alpha \geq 0$ untere Schranken für das betrachtete RCSP.

$$
\begin{aligned}
(\mathrm{LR}_{\mathrm{RCSP}}) \quad \min \quad & c^T x + \alpha^T (Rx - \lambda) \\
\text{s.t.} \quad & Nx = e_t - e_s \\
& x \in \{0,1\}^{|A|}
\end{aligned}
$$

Die beste untere Schranke, die wir so gewinnen können und fortan als Optimum der Lagrange Relaxierung bezeichnen wollen, ist somit

$$z = \max_{\alpha \geq 0} z_{LR}(\alpha).$$

Dieses Problem lässt sich mit Hilfe von Subgradientenverfahren allgemein lösen. Ansätze, die die spezielle Struktur des RCSP ausnutzen, werden im nächsten Abschnitt erläutert.
Des Weiteren sei mit d_{ij} der Optimalwert des $\mathrm{LR}_{\mathrm{RCSP}}$ mit Quelle i und Senke j bezeichnet. Dann folgt in azyklischen Graphen, dass d_{ij} gleich der Länge eines kürzesten Weges mit Kostenvektor $(c + \alpha R)$ abzüglich der Konstante $-\alpha^T \lambda$ ist. Dadurch kann man mit UB[1] $\in \mathbb{R}$ Knoten $k \in V$ aus dem Digraphen eliminieren [2], für die gilt:

$$d_{ik} + d_{kj} - \alpha^T \lambda > UB$$

[1] UB bezeichnet hierbei eine beliebige obere Schranke des RCSP.
[2] Dieses Vorgehen ist in [BC89] dokumentiert und wird allgemein zum Preprocessing genutzt.

2.4 Algorithmische Lösungsansätze

2.4.1 Dynamische Programmierung

Der im letzten Abschnitt gezeigte Zusammenhang zum KP liefert auch schon einen möglichen Lösungsansatz für das RCSP. Die Standardtechniken zum Lösen des KP sind auf der Idee der dynamischen Programmierung basierende Skalierungs- und Rundungsalgorithmen bzw. Approximationsschemata.
Die ersten Arbeiten hierzu sind [J66] von Joksch und [L76] von Lawler, die sich jedoch auf den Fall einer Ressource beschränken. Trotzdem sollen an dieser Stelle diese Arbeiten und ein voll polynomiales Approximationsschema von Hassin (92) vorgestellt werden. Dies soll darlegen, dass einfache pseudo-polynomiale Lösungansätze für den Ein-Ressourcen-Fall des RCSP existieren und diese mit Rundungs- und Skalierungstechniken zu *voll polynomialen Approximationschemata* [3] (FPAS) führen.

Rekursion nach Joksch (66) und Lawler (76)

Die grundlegende Idee der beiden Arbeiten von Joksch und Lawler beruht auf einer gültigen Rekursionsgleichung. Dazu sei das RCSP mit $k = 1$ sowie sowohl positiv ganzzahligen Ressourcenverbrauch, als auch positiv ganzzahligen Kosten gegeben. Dann sei ein Pfad vom Knoten i zum Knoten j ein r-Pfad, wenn dieser höchstens r Einheiten der Ressource verbraucht. Gemäß dieser Definition ist eine optimale Lösung des RCSP ein bzgl. der Kosten minimaler λ-Pfad vom Knoten 1 zum Knoten n. Seien weiterhin mit $c_j(r)$ die Kosten eines kürzesten r-Pfades von 1 zu j bezeichnet, dann kann man folgende gültige Rekursionsgleichung formulieren.

$$c_j(r) = \min\{c_j(r-1), \min_{(i,j)\in E, r_{ij}\leq r}\{c_i(r-r_{ij}) + c_{ij}\}\}$$

Abbildung 2.4: Rekursionsgleichung von Joksch und Lawler.

Scheinbar ist das Optimum des RCSP durch den Wert $c_n(\lambda)$ gegeben, welcher mittels obiger Gleichung rekursiv berechnet werden kann. Dazu sei $c_1(r) = 0$ für $0 \leq r \leq \lambda$, was aufgrund der positiven Bogenkosten plausibel ist. Andererseits setzt man $c_j(0) = \infty$ für die Knoten $j = 2, \ldots, n$ und erhält somit eine wohldefinierte, berechenbare Rekursion des RCSP.
Die formale Idee dahinter ist, herauszufinden, wie weit man mit einem um eine Einheit höherem Ressourcenverbrauch kommt bzw. wie man kürzeste l-Pfade mit $l \leq r - 1$ zulässig zu kürzesten r-Pfaden verlängern kann. Somit kann man $c_n(\lambda)$ in $O(|E|\lambda)$ berechnen. Dies ist zumindestens pseudopolynomial in der Kodierungslänge des Inputs, woraus direkt folgt, daß das RCSP schwach $\mathcal{NP}$-vollständig ist.
Analog zum KP kann man mit Hilfe von Skalierungs- und Rundungstechniken diese Ideen verwenden, um zu einem voll polynomialen Approximationsschema zu gelangen. In der Arbeit [H92] wurde dies diskutiert und umgesetzt.

[3] Eine genaue Definition dieser Klasse von Approximationsalgorithmen befindet sich im Anhang.

FPAS von Hassin (92)

Das folgende Approximationsschema von Hassin setzt ebenfalls voraus, dass es sich sowohl bei dem Verbrauch als auch bei den Kosten der Bögen um positive ganzzahlige Werte handelt und dass der gegebene Digraph azyklisch ist. Dann benutzt er folgende rekursive Idee:
Sei $g_j(c)$ der Ressourcenverbrauch eines kostenminimalen $(1,j)$-Pfades, dessen Kosten höchstens c betragen. Für diesen Wert gilt folgende Rekursionsformel:

$$g_j(c) = \min\{g_j(c-1), \min_{(i,j)\in E, c_{ij}\leq c}\{g_i(c-c_{ij}) + r_{ij}\}\}$$

Abbildung 2.5: Rekursionsgleichung von Hassin.

Startet man nun mit $g_1(c) = 0$ für $c = 0, 1 \ldots OPT$ und $g_j(c) = \infty$ für $j = 2, 3 \ldots, n$, so lässt sich das Optimum über obige Rekursionsgleichung berechnen. Zwar ist der Wert für OPT nicht bekannt, aber es gilt sicherlich $OPT = \min\{c \in \mathbb{N} \mid g_n(c) \leq \lambda\}$. [4]
Nun wird $g_j(c)$ zuerst für $c = 1$ und $j = 2, \ldots, n$ berechnet, danach für $c = 2$ und dies solange bis der erste Pfad gefunden wurde, der den Knoten n erreicht und für den $g_n(c) \leq \lambda$ gilt. Somit kann $OPT = c$ gesetzt werden, denn damit hat man den zulässigen $(1,n)$-Pfad gefunden, der $g_n(c)$ Ressourceneinheiten verbraucht und c Einheiten kostet. Da alle nicht betrachteten Pfade mindestens genauso teuer sind, hat man das Optimum generiert.
Algorithmisch muss man sich nur für jeden Knoten im Digraphen merken, welcher Knoten i bei der Minimumsbildung der Rekursion zu einer Aktualisierung geführt hat. Somit lässt sich der Pfad dann eindeutig rekonstruieren.
Man kann sich den Algorithmus als Tabelle mit dem entsprechenden Wert $g_j(c)$ vorstellen, wobei die Zeilen die Knoten und die Spalten die inkrementierten Kosten sind. So ist nur der Eintrag $g_n(OPT)$ von Interesse, der jedoch wieder rekursiv durch die ersten OPT-Spalten ermittelt werden kann. Um die Werte jeder Spalte korrekt zu ermitteln, braucht man für die Minimumsbildung konstant viel Zeit, jedoch muss man hierfür jeden Bogen genau einmal betrachten. Dies führt also zu einer Laufzeit von $O(OPT|E|)$.
Im Prinzip verfolgt Hassin dieselbe Idee wie Joksch und Lawler. Er vertauscht dabei nur die Rolle von Ressourcenverbrauch und Kosten.
Es bleibt noch zu erläutern, wie daraus ein FPAS konstruiert werden kann. Dazu sei T ein Testwert und $0 < \epsilon < 1$ die Konstante für das voll polynomiale ϵ − Approximationsschema. Mit diesen Parametern werden die Kosten c_{ij} wie folgt gerundet:

$$\overline{c}_{ij} = \lfloor \frac{c_{ij}(n-1)}{T\epsilon} \rfloor \cdot \frac{T\epsilon}{n-1}$$

Es gilt offenbar:

$$c_{ij} - \frac{T\epsilon}{n-1} < \overline{c}_{ij} \leq c_{ij}$$

[4] Andernfalls würde kein Pfad im Digraphen die Ressourcenbedingung erfüllen.

Daraus folgt, dass sich die Kosten jedes Pfades höchstens um $T\epsilon$ verringern. Nun kann man diese RCSP-Instanz mit skalierten Kosten $\lfloor \frac{\overline{c}_{ij}}{(T\epsilon/(n-1))} \rfloor$ durch den beschriebenen Algorithmus lösen und erhält ein $\overline{c}$ mit $g_n(\overline{c}) \leq \lambda$. Für den Optimalwert OPT der ursprünglichen Instanz lassen sich somit folgende Aussagen treffen:

- 1.Fall
$$\overline{c} < \frac{n-1}{\epsilon}$$
Dann gilt nach der Rücktransformation der Skalierung und Rundung für diesen λ-Pfad folgende Aussage:
$$\frac{T\epsilon}{n-1}\overline{c} + T\epsilon < T(1+\epsilon) \Rightarrow OPT < T(1+\epsilon)$$
- 2.Fall
$$\overline{c} \geq \frac{n-1}{\epsilon}$$
Wenn also jeder λ-Pfad mindestens Kosten $\geq \frac{n-1}{\epsilon}$ verursacht, gilt für die Kosten c jedes λ-Pfades nach der Rücktransformation der Kosten:
$$c \geq T \Rightarrow OPT \geq T$$

Damit ist garantiert, dass entweder $OPT \geq T$ oder $OPT < T(1+\epsilon)$ gilt. Durch das Runden und Skalieren der Bogenkosten kann man also den Problemterm OPT durch den Term $\log(\frac{n}{\epsilon})$ ersetzen und kommt für den Test zu einer benötigten Laufzeit von $O(|E|\log(\frac{n}{\epsilon}))$.
Somit kann man für ein festes ϵ und unteren (LB) und oberen Schranken (UB) für den Optimalwert OPT ein FPAS folgendermaßen konstruieren:
Man starte eine binäre Suche im Intervall (LB, UB) mit Hilfe der Testroutine und aktualisiere die Schranken, bis $UB \leq (1+\epsilon)LB$ gilt. Somit ist UB eine ϵ-approximative Lösung der gegebenen RCSP-Instanz. Abschließend ist festzuhalten, dass man mit Hilfe des FPAS von Hassin in $O(c|E|\log(\frac{n}{\epsilon}) + c)$ eine ϵ-approximative Lösung des RCSP bestimmen kann, wobei c eine von den initialen Schranken UB und LB abhängige Konstante sei.

Für diese Arten der Rekursionsgleichungen stellt es in azyklischen Graphen kein Problem dar, dass Bögen mit negativen Kosten existieren, denn sowohl die Rekursionsverankerung als auch die Rekursionsgleichung gelten weiterhin. Das Problem des Ansatzes der dynamischen Programmierung bei mehreren Ressourcen ist die Größe des Zustandsraumes und die Gefahr der 'Enumeration' aller Pfade. Bei einer Ressource, wie oben vorgestellt, scheint dies noch handhabbar.
Verfahren, die diese Ideen erweitern und auch für mehrere Ressourcen anwendbar machen, tauchen meist unter dem Namen *node labelling approach* auf. Jeder Knoten v besitzt eine Menge von Markierungen, die mit (s,v)-Pfaden korrespondieren. Diese sogenannten *Labels* enthalten als Information die Kosten und den Verbrauch der einzelnen Ressourcen des jeweiligen Pfades. Es werden jedoch ausschließlich Labels verwendet, die nicht von anderen dominiert werden. Der Begriff Dominanz bedeutet hier, ein (s,v)-Pfad P_1, bzw. sein Label, dominiert einen anderen (s,v)-Pfad P_2, bzw. sein Label, genau dann, wenn sowohl die Kosten als auch

alle Ressourcenverbrauchswerte von P_1 kleiner oder gleich den Werten von P_2 sind. All diese nicht dominierten Labels werden dann sukzessive berechnet. Dazu sind zu Beginn, bis auf die Labelmenge des Knoten s, die mit $0 \in \mathbb{R}^{1+k}$ initialisiert wird, alle Labelmengen leer.
Solange es unbearbeitete Labels gibt, wird ein Knoten mit einem solchen ausgewählt. Für diesen Knoten werden alle Pfade, die durch ausgehende Bögen entstehen können, auf ihre Ressourcenzulässigkeit hin überprüft und ggf. neue, nicht dominierte Labels erzeugt.
Diese werden dann der Labelmenge des entsprechenden Zielknotens des Bogens hinzugefügt. Das Label mit minimalen Kosten aus der Labelmenge des Knoten t liefert dann einen kürzesten ressourcenbeschränkten (s,t)-Pfad. Sollte die Labelmenge des Knoten t am Ende des Algorithmus leer sein, so ist gezeigt, dass kein ressourcenbeschränkter (s,t)-Pfad in diesem Digraphen existiert.
Der hier nur kurz informell vorgestellte Algorithmus stammt von Desrochers und Soumis [DS92], wobei im Ein-Ressourcen Fall mit Nicht-Null-Gewichten eine pseudo-polynomiale Laufzeit von $(|E|\lambda)$ nachgewiesen wurde.

2.4.2 k-th Shortest Path-Ansätze

Eine weitere Herangehensweise zum Lösen des RCSP sind k-th Shortest Path-Ansätze oder auch Path Ranking-Ansätze. Diese Verfahren beruhen auf der Idee, die Pfade aufsteigend bzgl. der Kostenfunktion zu sortieren und den ersten zulässigen Pfad als Lösung auszugeben.
Der größte Vorteil solcher Algorithmen ist, dass bei der Zulässigkeitsprüfung des Pfades nicht nur lineare Bedingungen überprüft werden können, sondern auch andere gewünschte Pfad- bzw. Teilpfadeigenschaften.
Im Allgemeinen werden bei solchen Algorithmen neue Bögen und Knoten im Digraphen generiert. Ist jedoch die Anzahl der zu konstruierenden Pfade, bevor man einen zulässigen Pfad gefunden hat, sehr hoch, dann kann dies durchaus zu exponentiellem Aufwand in der Kodierungslänge des urspünglichen Digraphen führen. Da dies bei den von uns bearbeiteten Instanzen aufgrund der Restriktivität der Resource Constraints der Fall ist, wird an dieser Stelle kein nur auf dieser Idee beruhender Algorithmus vorgestellt und anaylsiert.

2.4.3 Zwei-Phasen-Algorithmen

Die bereits analysierten Techniken haben gezeigt, dass eine Vielzahl von Ideen und Ansätzen existieren, jedoch sowohl theoretisch als auch praktisch keine annehmbare Laufzeit gewährleistet werden kann. Nur eine geeignete Kombination von diesen Techniken kann zu anwendbaren Algorithmen führen. Die Idee von Zwei-Phasen-Algorithmen für das RCSP ist folgende:

- In der ersten Phase werden untere Schranken und obere Schranken für das RCSP generiert, indem die Lagrange Relaxierung bzw. die LP Relaxierung 'optimal' gelöst werden. Die so gewonnenen Lagrange-Multiplikatoren, unteren und oberen Schranken können sowohl für Reduktion der Größe des Graphes, als auch für die Phase Zwei genutzt werden.

- In der zweiten Phase wird die mögliche Dualitätslücke geschlossen. Dazu werden verschiedene bereits vorgestellte Techniken verwendet.

Handler und Zang 1980

Die erste Veröffentlichung, welche diese Art von Lösungsansatz für das RCSP dokumentiert, ist [HZ80] 'A Dual Algorithm for the Constrained Shortest Path Problem' von G. Handler und I. Zang. Sie betrachten den Ein-Ressourcen-Fall und beschreiben explizit, wie sie die dazugehörige Lagrange Relaxierung optimal lösen. Dazu benutzen sie kombinatorische Methodik, um das Optimum der stückweise linearen, konvexen Funktion zu berechnen, wobei lediglich Berechnungen von kürzesten Wegen im Graphen mit transformierten Kosten durchgeführt werden müssen. In der zweiten Phase benutzen sie einen k-shortest Path Algorithmus, wobei die Kosten der Pfade gemäß der optimalen Lagrange-Multiplikatoren berechnet werden. Zur Laufzeit ihres Algorithmus haben sie jedoch weder für Phase Eins noch für Phase Zwei Aussagen bewiesen. Lediglich die Konvergenz gegen das Optimum nach endlich vielen Schritten haben sie gezeigt.

Beasley und Christofides 1989

Beasley und Christofides [BC89] führten diese Vorgehensweise fort. Sie betrachteten das RCSP mit mehreren Ressourcenbedingungen und nutzten deshalb Subgradientenverfahren, um die Lagrange Relaxierung approximativ zu lösen.
Für die zweite Phase benutzten sie eine Prozedur, die im gewissem Sinne mit Hilfe der Langrange-Multiplikatoren einen Branch&Bound&Cut-Baum generiert.

Mehlhorn und Ziegelmann 2000

In der Arbeit [MZ00] wurde letztendlich bewiesen, dass die von G. Handler und I. Zang beschriebene erste Phase ihres Algorithmus für den Ein-Ressourcen-Fall eine polynomiale Laufzeit besitzt. Mehlhorn und Ziegelmann formulierten eine, dank einer neuen detaillierten geometrischen Interpretation, für mehrere Ressourcen gültige Verallgemeinerung des Algorithmus, die sie mit *hull approach* bezeichnen.
Im Ein-Ressourcen-Fall ist das Vorgehen dabei analog zu dem von G. Handler und I. Zang. Für mehrere Ressourcen konnten sie noch keine Laufzeitabschätzung beweisen, lediglich theoretisch steht dank der Ellipsoidmethode fest, dass man die LP-Relaxierung des RCSP mit beliebig vielen Ressourcen unter gewissen Annahmen in polynomialer Zeit lösen kann. Diese Voraussetzungen wurden bereits im Abschnitt über die LP Relaxierung genau erläutert.
Mehlhorn und Ziegelmann erhalten in der ersten Phase durch ihren Hull Approach bessere Schranken als Beasley und Christofides durch ihre approximativen Subgradientenlösungen. Dadurch wird das Problem in Phase Zwei erheblich vereinfacht. Des Weiteren verwenden sie verschiedene neue Varianten, um die Dualitätslücke zu schließen.
Zum einen werden die Pfade gemäß der reduzierten Kosten mit Hilfe von k-Shortest Path Algorithmen durchlaufen (enumeriert). Zum anderen wurde ein geeignetes node labelling Verfahren vorgestellt, welches die ermittelten Schranken und reduzierten Kosten verwendet. Dieser informell vorgestellte Algorithmus stellt derzeit wohl die geschickteste Kombination aller Ideen dar und führt zu einem in der Praxis anwendbaren Lösungsalgorithmus für RCSP-Instanzen.

Kapitel 3

Das Dienstplanungsproblem

3.1 Modellierung

3.1.1 Graphentheoretisches Modell

Das Dienstplanungsproblem wird mit Hilfe des folgenden Digraphen modelliert:
Für alle Dienstelementbestandteile, deren Menge wir fortan mit V_D abkürzen wollen, und alle möglichen Ergänzungselemente der Menge V_E wird genau ein Knoten v konstruiert. Man beachte hierbei, dass erst a posteriori feststeht, welche Ergänzungselemente abgedeckt werden müssen. Diese Knoten enthalten alle sinnvollen Informationen und Attribute, die bereits in der Einleitung Erwähnung fanden. Das sind zum einen die Start-, End-, Lenk-, Arbeits- und mögliche Pausenzeit und zum anderen eine Vielzahl von Zulässigskeitsinformationen. [1]
Zwischen diesen Knoten existiert im Digraphen genau dann ein Bogen, wenn es eine Verknüpfung zwischen den zugrundeliegenden Elementen gibt, d.h. es könnte ein Dienst existieren, in dem diese Elemente aufeinander folgen. Vervollständigt wird dieses Netzwerk von einer Quelle und einer Senke. Diese Modellierung benötigt also folgende Mengen:

- Dienstelementbestandteile $v_d \in V_D = \{1, 2, 3 \dots n\}$
- Ergänzungelemente $v_e \in V_E = \{n+1, n+2 \dots, n+m\}$
- Dienstarten $dt \in \mathcal{DT} = \{1, 2 \dots, k\}$

Sei nun $D = (V, A)$ der Dienstplanungsgraph mit der Knotenmenge

$$V = V_D \cup V_E \cup \{0, n+m+1\}$$

und der Bogenmenge

[1] Auf die konkrete Einführung dieser Parameter und deren Bezeichnungen wird an dieser Stelle der Einfachheit halber verzichtet.

$$
\begin{array}{lll}
A = & \{(i,j) \in V \times V & \mid \text{Verknüpfung zwischen i und j existiert}\} \cup \\
& \{(0,j) \in V \times V & \mid \text{Element j ist als Dienstbeginn zulässig}\} \cup \\
& \{(i, m+n+1) \in V \times V & \mid \text{Element i ist als Dienstende zulässig}\}.
\end{array}
$$

Aus der Konstruktion der Bögen gemäß den Verknüpfungsregeln ergeben sich bereits folgende Bogentypen, deren Unterscheidung sich später als hilfreich herausstellen wird, und die deshalb hier Erwähnung finden sollen.

- Typ I
  Zwangsverknüpfungen sind diejenigen Bögen (u, v) zwischen zwei Knoten $u, v \in V_D$, bei denen v der einzige mögliche Nachfolger von u auf einem $(0, m+n+1)$-Pfad ist. Man beachte, dass dies erst nach Einlesen des kompletten Dienstplanungsgraphen ermittelt werden kann.

- Typ II
  Verknüpfungen innerhalb eines Dienststückes sind diejenigen Bögen (u, v) zwischen zwei Knoten $u, v \in V$, für die gilt, dass die Elemente u und v in einem Dienststück unmittelbar aufeinanderfolgend abgearbeitet werden können.

- Typ III
  Verknüpfungen für Dienststückwechsel sind Bögen (u, v) zwischen zwei Knoten $u, v \in V$, für die folgendes gilt:
  Wenn u und v aufeinanderfolgend abgearbeitet werden, findet ein Fahrzeugverlassen bzw. -wechsel statt und somit beginnt ein neues Dienststück. Der Knoten u beendet in diesem Fall das vorangegangene Dienststück, während der Knoten v das nächste einleitet. Bögen dieses Types können eine Unterbrechungszeit aufweisen, da u und v nicht unmittelbar zeitlich aufeinander folgen müssen.

- Typ IV
  Verknüpfungen für Dienstteilbeginn bzw. -ende sind Bögen zwischen der Quelle 0 bzw. der Senke $n+m+1$ und $v \in V_D \cup V_E$.
  Der Bogen $(0, v)$ symbolisiert dabei den Beginn des Dienstes, wobei v das erste abzuarbeitende Element darstellt. Diese Art von Bögen bilden die Bogenteilmenge $A_{IV_B} \subset A$ und werden daher fortan mit Typ IV_B bezeichnet.
  Ein Bogen $(v, n+m+1)$ stellt das Ende eines Dienstes mit v als letztem Element dar. Analog zur bereits eingeführten Notation werden diese Bögen mit Typ IV_E Bögen deklariert und sind daher Elemente der Menge $A_{IV_E} \subset A$.
  Man beachte hier zum einen, dass es sich bei der Menge A_{IV} um die disjunkte Vereinigung von Typ IV_B Bögen und IV_E Bögen handelt, und zum anderen, dass jeder Bogen $a \in A_{IV_B}$ nicht nur einen neuen Dienstteil sondern auch ein neues Dienststück einleitet.

Der Bogen (i, j) stellt also die lokale Kombinierbarkeit der Elemente dar, d.h. Element j könnte in einem Dienst nach dem Element i durchgeführt werden. So ist das Duty Scheduling Problem (DSP) in diesem Modell das Finden einer Menge von Diensten, die alle Dienstelemente aus V_D genau einmal überdeckt und bzgl. einer Kostenfunktion optimal ist.
Alle Dienste entsprechen in diesen Graphen also einem $(0, n+m+1)$-Pfad, andererseits sind nicht alle diese Pfade Dienste. Nur wenn der $(0, n+m+1)$-Pfad einer Dienstart $dt \in \mathcal{DT}$

zuzuordnen ist, handelt es sich um einen zulässigen Dienst.
Die folgenden drei Abbildungen sollen das eben eingeführte graphentheoretische Grundmodell bildlich untermauern.
Die Abbildung 3.1 stellt dabei einen kompletten Dienstplanungsgraph dar. Diese kleine Instanz ist aufgrund der Restriktivität der Dienstart, in diesem Fall der Teilzeitdienste, noch visualisierbar und bietet daher die Möglichkeit, alle Typen der Knoten und Bögen darzustellen.
Die Rolle der Ergänzungselemente wird in der Abbildung 3.2 beispielhaft für ein Dienstelement ebenso verdeutlicht, wie die Möglichkeit die Knoten topologisch gemäß der Startzeiten zu sortieren.
Abschließend werden in 3.3 exemplarisch Dienste im Dienstplanungsgraph 3.2 vorgestellt. Vor allem das Zusammenspiel zwischen den Typ III bzw. IV Bögen und den Ergänzungselementen soll hierbei aufgezeigt werden. Dabei handelt es sich beim ersten abgebildeten zulässigen Pfad um einen zusammenhängenden Dienst, der aus drei Dienststücken besteht.
Bei Dienst Zwei und Drei findet hingegen kein Fahrzeugwechsel statt. Diese Visualisierungen sollen bereits auf die Problematik hinweisen, dass keineswegs jeder $(0, n + m + 1)$-Pfad im zugrundeliegenden Digraphen einem zulässigen Dienst entspricht.

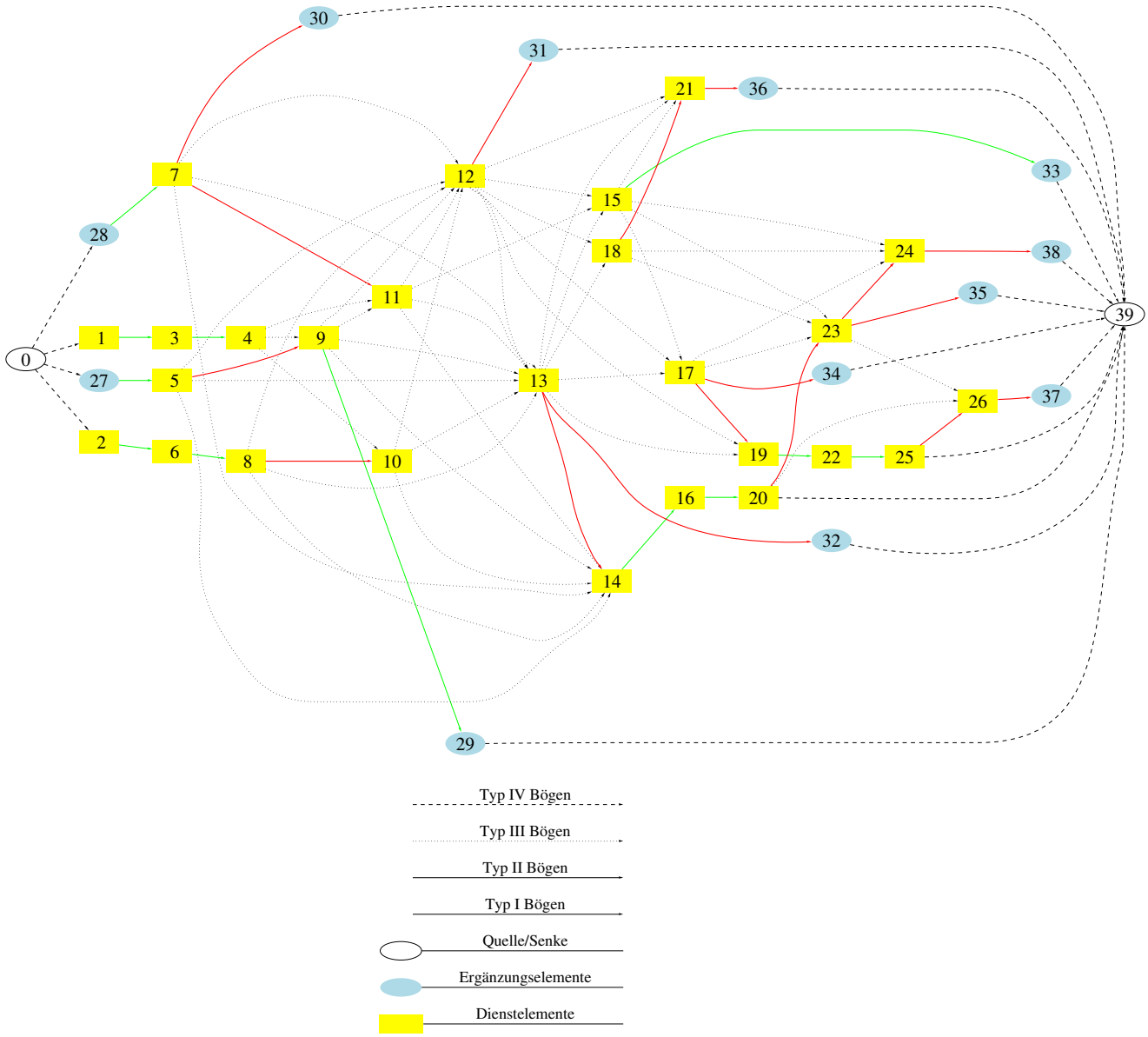

Abbildung 3.1: Visualisierung eines Dienstplanungsgraphen für Teilzeitdienste ($\mathcal{TZD}$).

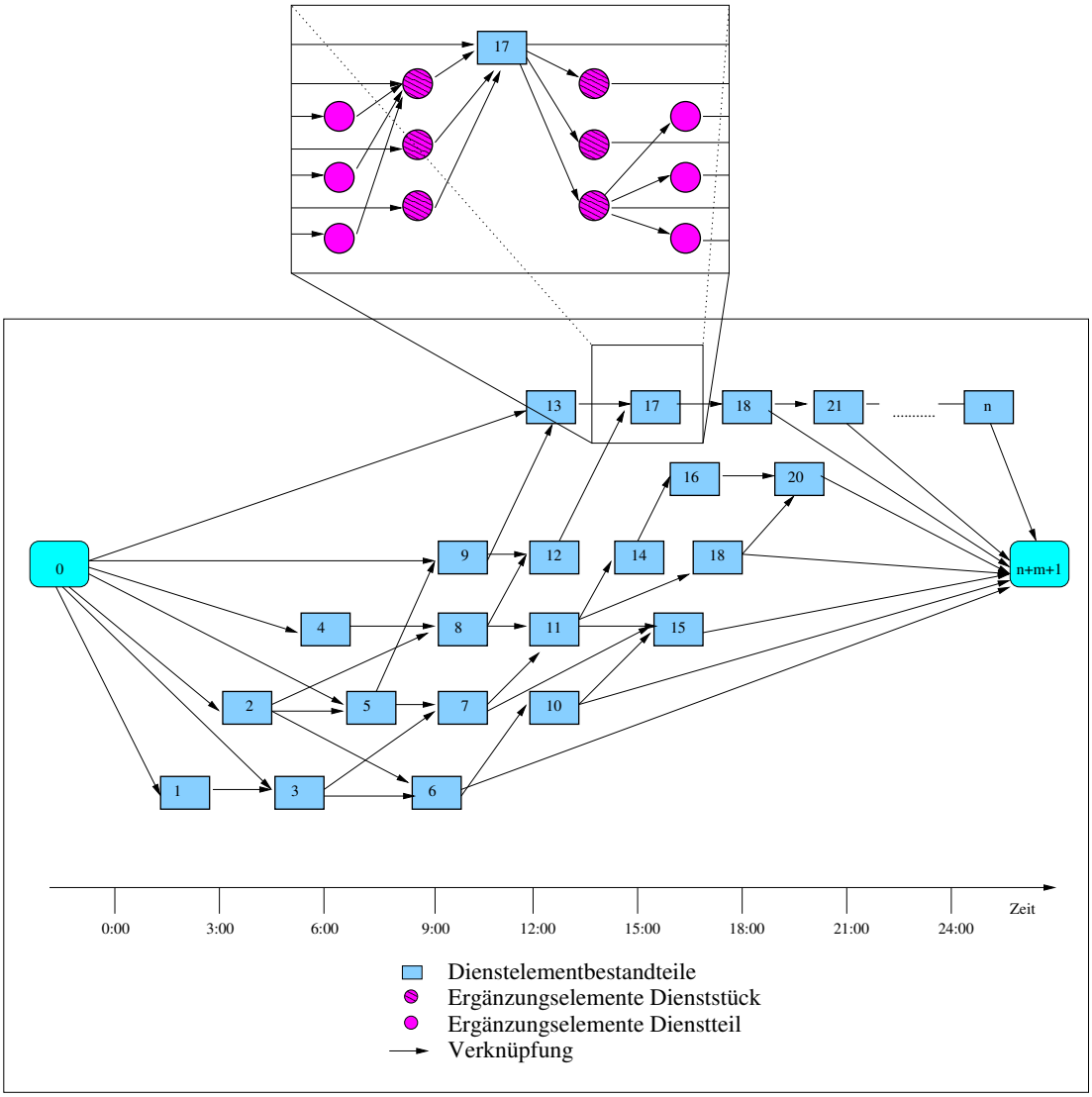

Abbildung 3.2: Dienstplanungsgraph und Umgebung der Dienstelemente.

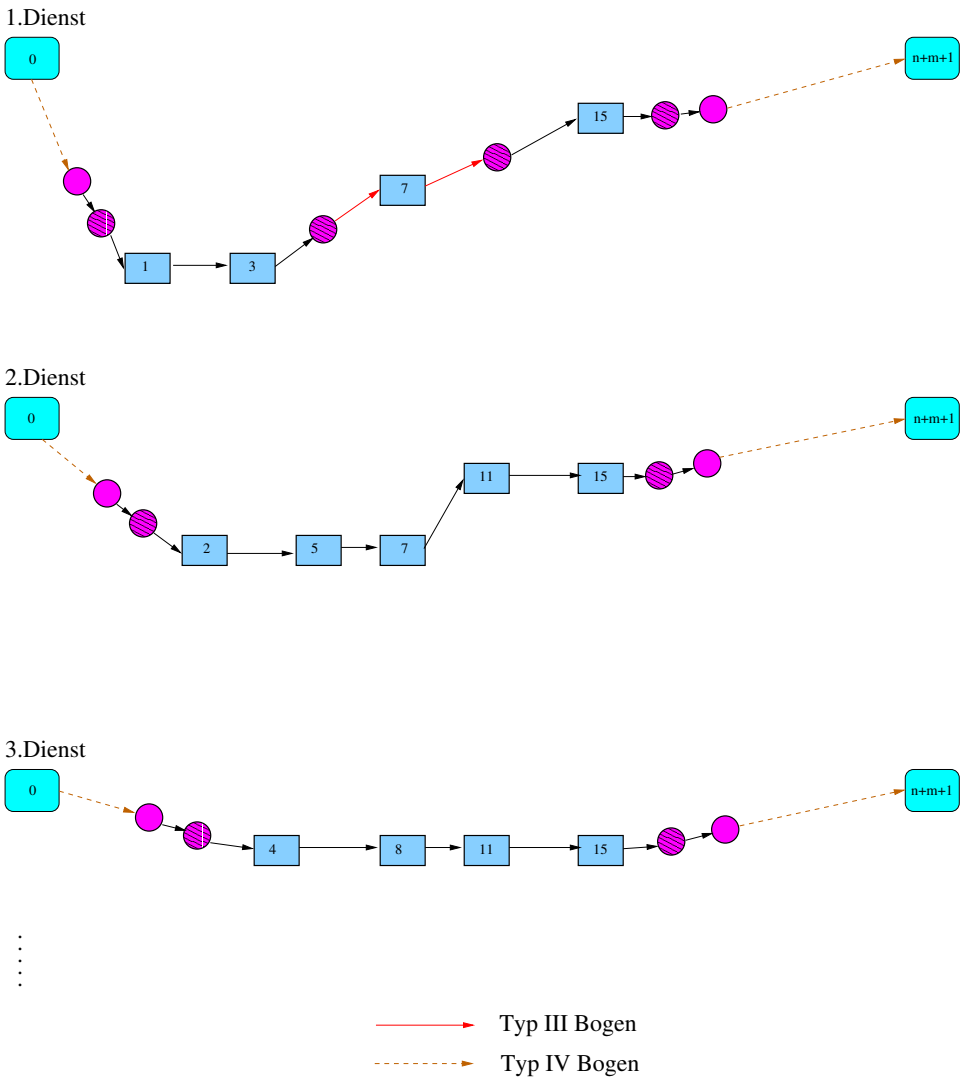

Abbildung 3.3: Menge von zulässigen Diensten.

3.1.2 Ursprüngliches Graphenmodell für mehrere Dienstteile

Aufgrund der Definition und Konstruktion der Bogenmenge des Graphen scheint es sinnvoll, den Graphen je nach Dienstart neu zu berechnen, da die Zulässigkeit der Verknüpfungen von den jeweiligen Parametern der Dienstart abhängt. Um im vorgestellten Graphenmodell sinnvoll geteilte Dienste darzustellen, werden zusätzliche Hilfsknoten $v \in V_H$ und weitere Bögen hinzugefügt. Diese Erweiterung des Modells soll hier erläutert werden. Zur Menge der Hilfsknoten wollen wir per Definition auch die Quelle 0 und die Senke $n + m + 1$ zählen.
Diese Hilfsknoten besitzen nur einen positiven Zeitparameter, der trivialerweise bei der Quelle Null und bei der Senke auf ein Maximum, z.B. die späteste Endzeit aller Elemente gesetzt werden kann. Für geteilte Dienste wird nun die Menge der Typ IV Bögen neu definiert. Jedes Element $v \in V_D \cup V_E$, welches ein zulässiges Startelement eines Dienstteiles darstellt, induziert einen neuen Hilfsknoten $h \in V_H$ und einen Typ IV_B Bogen (h, v).
Da für diese Betrachtung nur die Startzeiten der Knoten relevant sind und die Hilfsknoten diesen Parameter übernehmen, können mehrere Hilfsknoten mit gleicher Startzeit zu einem Hilfsknoten kontrahiert werden.
Analog ist die Vorgehensweise für diejenigen Knoten $u \in V$, die als letztes Element eines Dienstteiles zulässig sind. Hierbei wird jedoch der von diesem Knoten induzierte Hilfsknoten h mit der zulässigen Startzeit eines nächsten Dienstteiles parametrisiert. Der Zeitparameter des Knoten h ist somit die Summe aus der Endzeit des Knotens u und der Mindestunterbrechungsdauer der Dienstteile der jeweiligen Dienstart. Es wird also der Bogen $(u, h) \in A_{IV_E}$ konstruiert, und man erhält erneut eine disjunkte Partition der Menge A_{IV} in Dienstteilbeginn- und Dienstteilendbögen.
Um letztendlich das Netzwerk zu komplettieren, wird zwischen unmittelbar aufeinanderfolgenden Hilfsknoten ein Bogen hinzugefügt. Dazu werden die Hilfsknoten bzgl. des Zeitparameters aufsteigend sortiert. Man beachte hierbei, dass diese Zeitlinie mit der Quelle beginnt und mit der ursprünglichen Senke endet. [2] Um weiterhin auf einer disjunkten Partitionierung der Bogenmenge arbeiten zu können, sei für diese Bögen folgender Typ definiert:

- Typ V
  Hilfsverknüpfungen sind Bögen zwischen zwei Knoten $u, v \in V_H$. Dieser Bogentyp ist ausschließlich für die Modellierung der geteilten Dienste relevant.

[2] Durch spezielle weitere Parameter kann es sinnvoll sein diese Zeitlinie zu unterbrechen. Eine Bedingung für den spätesten Beginn des ersten Dienstteiles (oder des Dienstes allgemein) kann man beispielsweise integrieren, indem man den Typ V Bogen, der diese Zeitspanne abdeckt, entfernt.

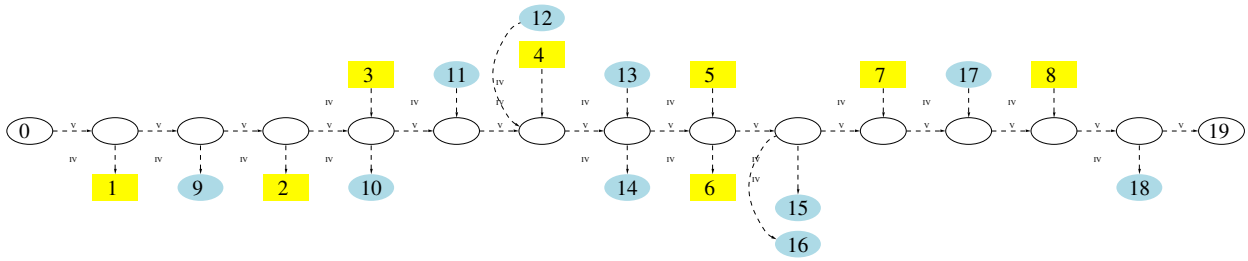

Abbildung 3.4: Dienstplanungsgraph mit Hilfsknoten, wobei nur Typ IV und V Bögen dargestellt sind.

Aus dieser Konstruktion folgen bereits für jeden Pfad von der Quelle zur Senke folgende trivialen Beobachtungen:

1. Die Anzahl der Typ IV Bögen eines $(0, n+m+1)$-Pfades ist gerade, wobei die ungeraden immer Typ IV_B und die geraden immer IV_E Bögen sind.
2. Auf solchen $(0, n+m+1)$-Pfaden können zwischen einem Typ IV_E Bogen und dem nächsten Typ IV_B Bogen ausschließlich Typ V Bögen liegen.

Ein Vorteil dieser Modellierung besteht darin, dass jedes mögliche Ende eines ersten Dienstteiles mit allen darauffolgenden Anfängen eines zweiten Dienstteiles kombiniert werden kann, ohne explizit für all diese Möglichkeiten einen Bogen zu konstruieren. Für reale Instanzen würden diese in der Anzahl der Knoten quadratisch vielen Kombinationen zu erheblich größeren Dienstplanungsgraphen führen, während die oben vorgestellte Modellierung mit Hilfsknoten linear in der Anzahl der Knoten bleibt. Dadurch wird der Lösungsraum, d.h. die (s,t)-Pfadmenge im Digraphen, möglicherweise unzulässig erweitert. Zwar ist durch diese Konstruktion bereits garantiert, dass die Mindestunterbrechungsdauer zwischen den Dienstteilen aufgrund der Definition der Typ IV_E Bögen eingehalten wird, hinsichtlich der maximalen Unterbrechungsdauer lassen sich jedoch keine impliziten Aussagen treffen. Es ist also möglich, (s,t)-Pfade zu konstruieren, die die maximale Unterbrechungsdauer überschreiten. Der Umgang mit derartigen Restriktionen in diesem Modell wird aber an anderer Stelle erläutert.

3.1.3 Dienstteilexpandiertes Graphenmodell

Die Motivation, die zu dieser neuen Art der Modellierung führte, wird an späterer Stelle erläutert. Hier soll ausschließlich die Modellierung anhand von zweigeteilten Diensten erklärt werden. Die Vorgehensweise für mehr als zwei Dienstteile ist zwar analog zu führen, jedoch sind solche Instanzen in der Praxis derart selten, dass wir an dieser Stelle zu Gunsten des Modellverständnisses ein Stück Allgemeinheit aufgeben können.
Jedes Dienstelement oder Ergänzungselement, welches sowohl im ersten als auch im zweiten Dienstteil erledigt werden könnte, wird zweimal (im allgemeinen für jeden möglichen Dienstteil), als Knoten v und v', repräsentiert. Aufgrund einiger Parameter der Dienstart, hier sind z.B. frühestes Ende des ersten Dienstteiles oder spätester zulässiger Beginn des zweiten zu nennen, existieren aber auch Knoten für die a priori feststeht, zu welchem Dienstteil sie gehören.
Nicht nur der Notation zuliebe erhält deshalb noch jeder Knoten und jeder Bogen einen Parameter, der den gegenwärtigen Dienstteil symbolisiert. Nun werden analog zum ursprünglichen Modell die Bogenmengen konstruiert. Falls also die Verknüpfung zwischen u und v für die betrachtete Dienstart existiert, wird zusätzlich zum Bogen (u, v) auch der Bogen (u', v') konstruiert, sofern es sich sowohl bei u als auch bei v um verdoppelte Knoten handelt. Alle möglichen Fälle sind hierbei denkbar, d.h. es können der Bogen (u', v'), der Bogen (u, v), beide , aber auch keiner der beiden existieren.
Danach wird die Menge der Typ IV Bögen, ebenfalls wie im ursprünglichen Modell, zur Bogenmenge hinzugefügt. Dabei bleiben alle Parameter der Bögen gleich, bis auf die Tatsache, dass jetzt je nach Typ der Start- bzw. Endknoten eine Dienstteilzuordnung für alle Bögen getroffen werden kann.
Zum Schluss wird noch mit Hilfe der Typ V Bögen das Netzwerk komplettiert. Per Definition gehören diese, ebenso wie die Hilfsknoten, zu keinem der beiden Dienstteile. Vorteil dieser Modellierung ist die nun mögliche Zuordnung zwischen den Bögen und den Dienstteilen [3]. Dadurch kann man Eigenschaften, die sich auf Dienstteile beziehen, besser und exakter modellieren als im ursprünglichen Modell. Für zwei Dienstteile bezahlt man diese Vorteile im schlechtesten Fall durch eine Verdopplung (im allgemeinen eine Vervielfachung um den Faktor k, wobei k der maximal zulässigen Anzahl Dienstteile der jeweiligen Dienstart entspricht) der Bogen- und Knotenmenge. Das durch dieses alternative Modell gewonnene Netzwerk wollen wir fortan als *dienstteilexpandierten Dienstplanungsgraph* bezeichnen.

[3] Dieses Problem stellt sich trivialerweise nicht bei zusammenhängenden, d.h. nicht geteilten, Diensten.

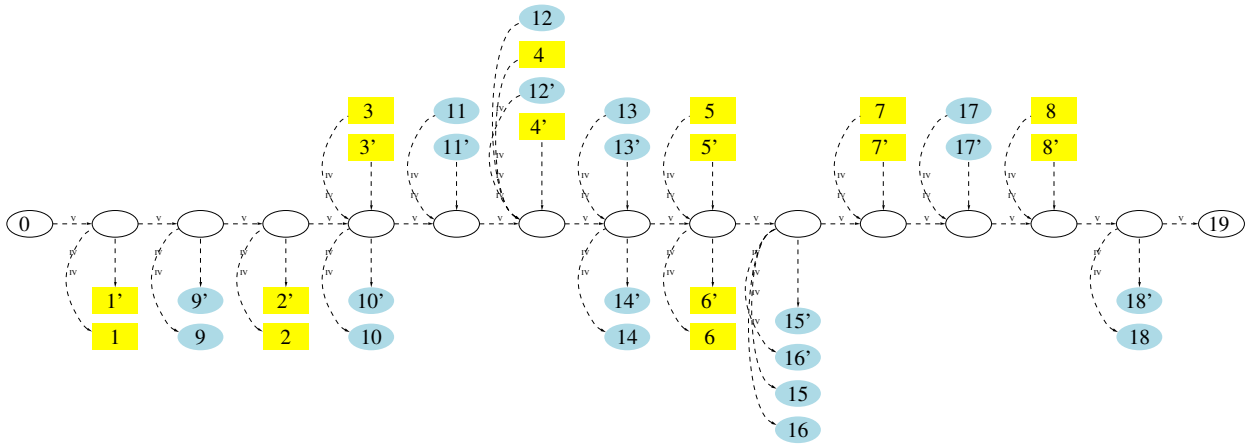

Abbildung 3.5: Dienstteilexpandierter Dienstplanungsgraph, wobei nur Typ IV und V Bögen dargestellt sind.

Durch diese Modellierung erhält man, zusätzlich zu der bisher bekannten disjunkten Partitionierung der Bogenmenge gemäß der Bogentypen $A = A_I \dot\cup A_{II} \dot\cup A_{III} \dot\cup A_{IV} \dot\cup A_V$, noch eine weitere hilfreiche Partitionierung nach Dienstteilen $A = A_1 \dot\cup A_2 \dot\cup A_H$. Beispielsweise enthält die Menge $A_I \cap A_{IV_B}$ alle zulässigen Bögen die den ersten Dienstteil einleiten.
Analog lässt sich für Dienstarten mit zwei Dienstteilen die Knotenmenge in $V = V_D \dot\cup V_E \dot\cup V_H$ bzw. $V = V_1 \dot\cup V_2 \dot\cup V_H$ partitionieren.
Ein kleiner Nachteil dieses dienstteilexpandierten Dienstplanungsgraphen für geteilte Dienste resultiert aus der Hilfskonstruktion für die Kombination der Dienstteile. Im ursprünglichen Modell wird die zeitlich korrekte Reihenfolge der Dienstteile aufgrund der Konstruktion des Graphen sichergestellt, während im dienstteilexpandierten Dienstplanungsgraphen darüber keine Ausssage getroffen werden kann. Insbesondere existieren unzulässige Pfade, die zuerst den zweiten Dienstteil einleiten und danach den ersten. Zwar können (aufgrund der Positionslogik der Ergänzungselemente müssen sie jedoch nicht) diese mit einem zulässigen Pfad korrespondieren, dennoch wird die konstruierbare Pfadmenge unzulässig erweitert.

3.1.4 Set Partioning Modell

Nachdem nun ausführlich das Graphenmodell vorgestellt wurde, soll in diesem Abschnitt das eigentliche Dienstplanungsproblem allgemein formuliert werden.
Sei die Menge aller Dienste mit $\mathcal{D}$ bezeichnet, x_d, $d \in \mathcal{D}$ die 0/1-Entscheidungsvariable, die angibt, ob Dienst d im Dienstplan enthalten ist, und $\omega \in \mathbb{R}_+^{|\mathcal{D}|}$ ein Vektor, der jedem Dienst seine spezifischen Kosten zuordnet, so kann das DSP mit Hilfe der Dienstelementbestandteil-Dienst-Inzidenzmatrix $\Phi \in \{0,1\}^{n \times |\mathcal{D}|}$ als Set Partioning Problem (SPP) formuliert werden.
Durch die bereits erwähnten Base Constraints können allgemein weitere lineare Anforderungen an den Dienstplan gestellt werden. Dies beinhaltet in der Praxis beispielsweise Restriktionen bzgl. der minimalen bzw. maximalen Anzahl der Dienste einer gewissen Dienstart. Sei beispielsweise mit $\mathcal{TZD} \subset \mathcal{D}$ die Menge aller Teilzeitdienste bezeichnet, so modelliert die Ungleichung $\sum_{d \in \mathcal{TZD}} x_d \leq t$ die Beschränkung der Anzahl verwendeter Teilzeitdienste auf maximal t .
Aber auch ein bevorzugter Dienstmix der Dienstarten oder Diensttypen, d.h. ein angestrebtes ausgewogenes Verhältnis, kann durch eine solche Ungleichung erzielt werden. Die Einträge $b_{ij} \in \mathbb{R}_+$ der Base Constraints Matrix $B \in \mathbb{R}_+^{k \times |\mathcal{D}|}$ müssen dementsprechend gewählt werden, wobei k der Anzahl der eingebauten Base Constraints entspricht. Ohne Beschränkung der Allgemeinheit sei für alle Base Constraints die rechte Seite der Ungleichung zu Eins normiert.
Zusammengefasst ergibt sich folgendes System :

$$\begin{aligned} \text{(SPP)} \quad \min \quad & \omega^T x \\ \text{s.t.} \quad & \Phi x = \mathbb{1} \\ & Bx \leq \mathbb{1} \\ & x \in \{0,1\}^{|\mathcal{D}|}. \end{aligned}$$

Optionale Dienstelemente können analog zu Ergänzungselementen betrachtet werden, d.h. an solche existieren keine expliziten Bedingungen. Man beachte jedoch, dass implizit durch die Menge der Dienste und deren Kosten die optionalen Dienstelemente und die Menge der Ergänzungselemente im Modell integriert sind.
Für den Fall, dass ein Dienstelementbestandteil i mehrmals überdeckt werden darf, wird einfach die Bedingung $\Phi_i.x = 1$ durch $\Phi_i.x \geq 1$ ersetzt. Bei Dienstplanungsproblemen dieser Art kann man somit die Menge der Dienstelementbestandteile in einen Set Partioning- und einen Set Covering-Teil aufspalten. Komplexitätsaussagen zum SPP werden in dieser Arbeit nicht ausführlich dargelegt. Hier wird auf [GJ79] verwiesen, in dem gezeigt wurde, dass das SPP $\mathcal{NP}$-schwer ist.

3.1.5 Constrained Path Partioning Modell

Dieser Abschnitt dient dazu, die Verbindung zwischen dem vorgestellten Graphenmodell und der Formulierung des DSP als SPP herzustellen. Dazu lässt sich das DSP analog als ein Constrained Path Partioning Problem (CPPP) formulieren, d.h. alle von einem Dienstelementbestandteil induzierten Knoten sollen mit einer Menge von zulässigen Pfaden im vorgestellten Dienstplanungsgraphen kostenminimal überdeckt werden. Auch dieses Modell benötigt 0/1-Entscheidungsvariablen für alle zulässigen Pfade, deren Menge fortan mit $\mathcal{FP}$ bezeichnet sei. Die bereits für das SPP eingeführten Parameter ω für die Dienste und B für die Base Constraints lassen sich problemlos übernehmen, so dass man letzendlich folgendes Modell mit $\chi \in \{0,1\}^{n \times |\mathcal{FP}|}$ als Dienstelementbestandteil-Pfad-Inzidenzmatrix erhält:

$$\begin{aligned} (\text{CPPP}) \quad \min \quad & \omega^T x \\ \text{s.t.} \quad & \chi x = \mathbb{1} \\ & Bx \leq \mathbb{1} \\ & x \in \{0,1\}^{|\mathcal{FP}|}. \end{aligned}$$

Sei nun $\mathcal{P}$ die Menge aller $(0, n+m+1)$-Pfade und somit $\mathcal{IP} = \mathcal{P} \backslash \mathcal{FP}$ die Menge der unzulässigen Pfade, so lässt sich später diese Modellierung einfacher analysieren und relaxieren. Dies wird deutlich, wenn man sich die bestehenden Lösungsalgorithmen für das SPP ansieht, was im folgenden Abschnitt nur skizzenhaft und informell geschieht. Die Formulierung des DSP als SPP unterscheidet sich mathematisch nicht von der Modellierung des DSP als CPPP. Diese Art der Modellierung dient vielmehr dazu, die Sichtweise der später aufgestellten Relaxierungen zu verstehen.

3.2 Lösungsansätze für das SPP

Um die algorithmischen Lösungstechniken vorzustellen, reicht es vorerst, das SPP ohne Base Constraints zu betrachten, da sich die Vorgehensweise kaum ändert. Dafür sind folgende Variablen- und Mengendefinitionen notwendig:
Sei $\mathcal{D} = \{1, 2, 3 \ldots q\}$ die Dienstmenge und $\overline{\mathcal{D}} \subseteq \mathcal{D}$ mit $|\overline{\mathcal{D}}| = \overline{q}$ eine noch genauer zu definierende Teilmenge. Des Weiteren sei $l \in \{0,1\}^q$ ein Vektor, der zum Festsetzen einzelner Dienste dient, d.h. wird l_d für $d \in \mathcal{D}$ Eins gesetzt, bedeutet dies, dass dieser Dienst in den betrachteten Dienstplänen bereits fixiert ist. Entsprechend der obigen Indexmengen seien $\overline{x}$, $\overline{l}$ und $\overline{\Phi}$, die auf diese jeweilige Menge $\overline{\mathcal{D}}$ beschränkten Vektoren und Matrizen.

$$(SPP) \qquad \min \omega^T x, \;\; \Phi x = \mathbb{1}, x \in \{0,1\}^{|\mathcal{D}|}$$

Ausgangspunkt ist das SPP, dessen Optimallösungen aufgrund der Modellierung mit optimalen Dienstplänen des DSP korrespondieren. Dennoch ist es utopisch, dieses ganzzahlige Programm

optimal lösen,geschweige denn formulieren zu wollen. Allein das Berechnen der Matrix Φ würde eine Aufzählung (Enumeration) aller möglichen Dienste bedeuten. Auch ein weiteres Arbeiten mit einer Matrix solcher Größe ist nicht vorstellbar, man beachte, dass hierbei soviele Spalten wie Dienste bzw. zulässige Pfade im Dienstplanungsgraph existieren. Deshalb versucht man gute untere Schranken für das Problem zu berechnen und betrachtet folgende Relaxierung:

$$(MLP) \qquad \min \omega^T x, \;\; \Phi x = \mathbb{1}, l \leq x, 0 \leq x \leq \mathbb{1}$$

Das MasterLP (MLP) ist hierbei die Standard LP Relaxierung des SPP, das (DLP) das zum (MLP) duale LP.

$$\begin{aligned} (DLP) \qquad & \max \pi^T(\mathbb{1} - \Phi l) + \omega^T l, \;\; \pi^T \Phi \leq \omega^T \\ (RMLP) \qquad & \min \overline{\omega}^T \overline{x}, \;\; \overline{\Phi}\overline{x} = \mathbb{1}, \overline{l} \leq \overline{x}, 0 \leq \overline{x} \leq \mathbb{1}, \\ (RDLP) \qquad & \max \pi^T(\mathbb{1} - \overline{\Phi l}) + \overline{\omega}^T \overline{l}, \;\; \pi^T \overline{\Phi} \leq \overline{\omega}^T \end{aligned}$$

Beim RMLP und RDLP handelt sich um eine auf ausgewählte Variablen beschränkte Version des MLP und deren duales Programm DLP, wobei $\pi \in \mathbb{R}^n$ die duale Variable ist. Das Optimum des MLP liefert eine untere Schranke für das SPP. Um jedoch das MLP optimal lösen zu können, ohne alle Dienste explizit zu betrachten (enumerieren), arbeitet man nur mit einer Teilmenge der Dienstmenge.
Die Initialisierung dieser Teilmenge kann zum Beispiel durch Generieren von Diensten (Enumeration) spezieller Dienstarten, durch Verwendung alter bekannter Dienste oder durch dynamisches Generieren von Pfaden im Dienstplanungsgraph erfolgen.
Die so gewonnenen RMLP und RDLP können nun mit Hilfe von LP-Solvern, in unserem Fall CPLEX 8.0, optimal gelöst werden. Unter der Annahme, dass wir die Menge $\overline{\mathcal{D}}$ sinnvoll gewählt haben, d.h., dass Lösungen existieren, kann man gemäß der linearen Optimierung und der Dualitätstheorie wie folgt argumentieren:
Sobald es keinen Dienst mehr gibt, dessen reduzierte Kosten (vgl. lineare Optimierung) negativ sind, handelt es sich bei der optimalen Lösung des RMLP bereits um eine optimale Lösung des MLP. Dazu betrachtet man folgende Fragestellung :

$$(PRICE) \;\; \exists j \in \mathcal{D} : \;\; \omega_j - \pi_\star^T \Phi_{.j} < 0$$

Diese gesamte Lösungstechnik wird als Column-Generation-Verfahren bezeichnet, d.h. es wird nur für eine beschränkte Anzahl von Diensten optimiert und dann versucht, Lösungen des PRICE Problemes hinzuzufügen. Die optimalen Lösungen des RMLP liefern offensichtlich obere Schranken für das Optimum des MLP, aber auch untere Schranken. Dies wird aber an späterer Stelle des Kapitels thematisiert und bewiesen. Insgesamt führt das zu einer beweisbaren unteren Schranke für das SPP, so dass es dann möglich ist, die durch Heuristiken gewonnenen Dienstpläne zu bewerten und die mögliche Differenz zum Optimum abzuschätzen.
In Abbildung 3.6 wird vereinfacht diese angewandte Verfahrensweise dargestellt. Im Großen und Ganzen soll dies nur übersichtsartig das Vorgehen beim Lösen von DSP-Instanzen vorstellen und aufzeigen, dass eine Menge von kleineren Unterproblemen wiederholt gelöst werden muss. Hierzu werden eine Vielzahl von optimierenden und optimierten Algorithmen, z.B. Subgradientenverfahren, Branch & Bound & Cut-Verfahren, speziellen Simplexalgorithmen, aber

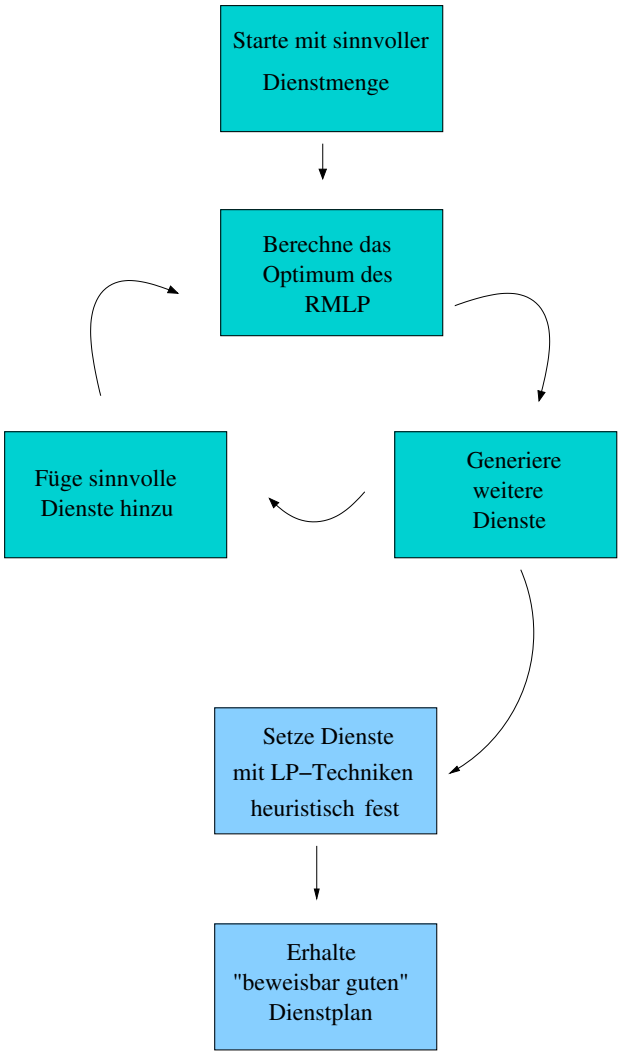

Abbildung 3.6: Algorithmische Idee des Column-Generation-Ansatzes für das DSP.

auch kürzeste Wege Algorithmen verwendet und mit Hilfe von heuristischen Ideen beschleunigt. Die Beweise und expliziten Vorgehensweisen des Algorithmus sind in [BGL01] dargestellt. Interressant und fundamental für diese Arbeit ist hierbei vor allem, dass das RCSP als ein Unterproblem beim Lösen des MLP auftaucht. Diese Zusammenhänge sollen Thema des nächsten Abschnittes sein.

3.3 PRICE und RCSP

Dieser Abschnitt soll nun die Möglichkeit darlegen, mit Hilfe des RCSP Aussagen über das bearbeitete DSP, genauer gesagt über das aktuelle RMLP, zu treffen. Möglich wird dies durch das folgende Zusammenspiel zwischen dem PRICE Problem und dem RCSP:
Angenommen die Dienstmenge $\mathcal{D}$ sei bereits durch Bedingungen der Art (3.3) und (3.4) definiert, so folgt nämlich eine gewisse Äquivalenz beider Probleme, d.h. die Frage nach der Existenz eines Dienstes mit negativen reduzierten Kosten lässt sich mit der Suche nach einem ressourcenbeschränkten kürzesten Weg beantworten. Zum einen folgt triviealerweise aus der Existenz eines solchen ressourcenbeschränkten kürzesten Weges mit negativen Kosten die Existenz eines Dienstes mit eben diesen negativen reduzierten Kosten. Zum anderen folgt, falls der ressourcenbeschränkte kürzeste Weg positive Kosten besitzt, dass es keine Lösung des aktuellen PRICE Problems geben kann.
Die Existenz und Berechenbarkeit eines solchen kürzesten Pfades verdankt man der Tatsache, dass es sich der Konstruktion nach bereits um ein azyklisches Netzwerk handelt. Dies ist leicht einzusehen, da die Knoten des Netzwerkes zu Arbeitseinheiten mit festen Startzeiten korrespondieren. Sortiert man nun diese Knoten aufsteigend nach der Startzeit, so erhält man eine topologische Knotensortierung des Digraphen, d.h. jeder Bogen im Digraphen D führt von einen Knoten mit topologisch kleineren Wert zu einem Knoten mit topologisch grösseren Wert. Die Existenz dieser topologischen Sortierung ist sowohl in der Praxis als auch in der Theorie äußerst hilfreich.
Denn aufgrund der Äquivalenz der Aussagen der 'Digraph ist zykelfrei bzw. azyklisch' und der 'Digraph besitzt eine topologische Sortierung' folgt obere Existenzaussage. Insbesondere existieren somit trotz der möglicherweise negativen Kosten der Bögen keine negativen Zykel, wodurch die Wohldefiniertheit des aufgestellten RCSP gezeigt ist. Diese Zusammenhänge wurden bereits in [BGL01] vorgestellt. Für die praktische Umsetzung und Implementierung bedeutet dies, dass man geeignete Datenstrukturen und Algorithmen verwenden kann, die dies zu Gunsten der Laufzeit ausnutzen. Gemäß der eingeführten Notation des letzten Abschnittes ergibt sich folgende mathematische Argumentation:

$$(PRICE)\ \ \exists j \in \mathcal{D}:\ \ \omega_j - \pi^T \Phi_{.j} < 0$$

$$\Longleftrightarrow$$

$$(PRICE)\ \ \exists j \in \mathcal{FP}:\ \ \omega_j - \pi^T \chi_{.j} < 0$$

$$\Longleftrightarrow$$

$$\begin{aligned} (PRICE') \quad \min \quad & \overline{c}^T x + \overline{d}^T s \\ \text{s.t.} \quad & x \in \operatorname*{conv}_{P \in \mathcal{FP}} (\chi_{\cdot P}) \\ & x \in \{0,1\}^{|A|} \end{aligned}$$

$$
\begin{aligned}
(RCSP) \quad \min \quad & \overline{c}^T x + \overline{d}^T s && (3.1)\\
\text{s.t.} \quad & Nx = e_{n+m+1} - e_0 && (3.2)\\
& Rx + s^+ - s^- = \lambda && (3.3)\\
& 0 \leq s \leq u && (3.4)\\
& x \in \{0,1\}^{|A|} && (3.5)
\end{aligned}
$$

Dazu sei die Zielfunktion des RCSP wie folgt definiert:

$$\overline{c}^T x + \overline{d}^T s \quad = \quad c^T x + d^T s - \sum_{j=1}^{n+m} \sum_{(i,j)\in A} \pi_j x_{(i,j)},$$

wobei $d = (d_+, d_-) \geq 0$ festgelegt wird, damit s^+ und s^- eindeutig sind. D.h. $s(x) = (s^+(x), s^-(x))$, und entweder gilt $s = (\lambda - Rx, 0)$ oder $s = (0, Rx - \lambda)$.
Mit Hilfe der Gleichungen (3.3) und (3.4) werden maximale, minimale und optimale lineare Bedingungen modelliert. Dazu sind die Schranken u für s geeignet zu wählen. Der Vektor d dient als Bestrafung (Penalty) für die erlaubte Abweichung vom angestrebten optimalen Ressourcenverbrauchsvektor λ.
Ein Problem in der Dienstplanung an praxisnahen Instanzen ist, dass sich Dienste nicht ausschließlich über Resource Constraints abgrenzen lassen. Hier sind vor allem Pausenregelungen und deren Anrechnungsregel, Positionslogik und Dienstteil- bzw. Dienststückeigenschaften zu nennen. Alle diese Anforderungen an die Dienste sind nicht-lineare Sachverhalte im Sinne unseres zugrundeliegenden ursprünglichen Graphenmodells.
Eine weitere Voraussetzung dieser gesamten Äquivalenzargumentation ist eine gewisse Modularität der Kosten. D.h. die Kosten eines Dienstes $j \in \mathcal{D}$, dessen Bogeninzidenzvektor des zum Dienst j korrespondierenden Pfades P fortan mit χ_P deklariert wird, müssen sich aus den Kosten der einzelnen Dienstelemente, Ergänzungselemente, Verknüpfungen und den Bestrafungskosten für zulässige Abweichungen der linearen Bedingungen zusammensetzen. Seien mit c_a die Kosten des Bogens $a = (i, j)$ und des Zielknoten j bezeichnet, dann muss gelten:

$$\omega_j = c^T \chi_P + d_+^T s^+(\chi_P) + d_-^T s^-(\chi_P) = \sum_{e\in P} c_e + d^T s(\chi_P)$$

Für die Fälle, in denen das nicht gewährleistet werden kann, liefert das Optimum des RCSP Modell nur eine untere Schranke für das zugehörige Minimierungsproblem PRICE'.
Denn offenbar gilt in diesem Fall, dass jeder Dienst eine zulässige Lösung des RCSP ist, aber keineswegs jede zulässige Lösung des RCSP, d.h. jeder ressourcenbeschränkte Pfad, einem zulässigen Dienst entspricht. Somit ist einleuchtend, dass, falls ein kostenminimaler ressourcenbeschränkter Pfad, d.h. eine Optimallösung des RCSP, einen positiven Zielfunktionswert hat, kein Dienst mit negativen Kosten existiert. Die gegenwärtige Optimallösung des RMLP ist in diesen Fall also auch optimal für das MLP.

Aber auch im ungünstigen Fall, d.h. wenn ein kostenminimaler ressourcenbeschränkter Pfad einen negativen Zielfunktionswert aufweist, will man Aussagen über den aktuelle Optimalwert des RMLP in Bezug auf das MLP treffen. Das dies möglich ist wird im folgenden Abschnitt bewiesen.

3.4 RCSP Lower Bound für das DSP

Im letzten Abschnitt ist kurz dargelegt worden, welche Techniken zum Lösen des MLP angewandt werden und welche Rolle dabei das PRICE Problem und das RCSP spielen. Es ist trivial, dass jede Lösung des RMLP für eine sinnvolle Menge $\overline{\mathcal{D}}$ eine obere Schranke für das Optimum des MLP liefert. Dabei ist mit sinnvoller Menge $\overline{\mathcal{D}}$ gemeint, dass das dazugehörige RMLP zulässige Lösungen besitzt, was in unserem konkreten Szenario durch sogenannte Tripper-Dienste (bzw. Pfade), die nur ein $v \in V_D$ abdecken und enorme Kosten verursachen, zweifellos gewährleistet ist. Betrachten wir nun deshalb folgende zueinander duale lineare Programme:

$$\begin{array}{llll} \text{(RMLP)} & \min & \overline{\omega}^T\overline{x} & \geq \\ & \text{s.t.} & \overline{\Phi}\overline{x} = \mathbb{1} & \\ & & \overline{x} \leq \mathbb{1} - \overline{l} & \\ & & \overline{x} \geq 0. & \end{array} \qquad \begin{array}{lll} \text{(RDLP)} & \max & \pi^T(\mathbb{1} - \overline{\Phi l}) + \overline{\omega}^T\overline{l} \\ & \text{s.t.} & \pi^T\overline{\Phi} \leq \overline{\omega}^T \end{array}$$

Angenommen wir haben nun eine optimale Lösung $\overline{x_\star}$ des RMLP und $\pi_\star$ des RDLP ermitteln können, dann gilt offenbar $z = \overline{\omega}^T\overline{x_\star} = \pi_\star^T(\mathbb{1} - \overline{\Phi l}) + \overline{\omega}^T\overline{l}$.
Falls für dieses $\pi_\star$ das PRICE Problem keine Lösung besitzt, so wäre $\overline{x_\star}$ auch optimal für das MLP.
Dies folgt direkt aus der Dualitätstheorie, da in diesem Fall $\pi_\star$ eine zulässige Lösung des DLP ist und den gleichen Zielfunktionswert wie $\overline{x_\star}$ aufweist. Man könnte analog auch mit Hilfe des Simplexalgorithmus und der Definition der zugrundeliegenden reduzierten Kosten argumentieren.
Für den anderen, wohl wahrscheinlicheren Fall, lassen sich mit Hilfe einer δ-Transformation der Kosten dennoch Aussagen und Schranken gewinnen. Betrachtet sei deshalb folgende Fragestellung:

$$(\delta - PRICE)\ \exists j \in \mathcal{D}: \ \ (1+\delta)\omega_j - \pi_\star^T\Phi_{\cdot j} < 0$$

Wählt man nun $\delta \in \mathbb{R}_+$ minimal, so dass das δ-PRICE Problem keine Lösung besitzt, und seien des Weiteren $x_\star \in \mathbb{R}^{|\mathcal{D}|}$ und $l \in \{0,1\}^{|\mathcal{D}|}$ wie folgt definiert:

$$(x_\star)_i = \begin{cases} (\overline{x_\star})_i, & i \in \overline{\mathcal{D}} \\ 0 \ \text{ sonst}, \end{cases}$$

$$(l)_i = \begin{cases} \overline{l}_i, & i \in \overline{\mathcal{D}} \\ 0 \text{ sonst.} \end{cases}$$

Betrachtet man jetzt die beiden folgenden δ-transformatierten LPs,

$$\begin{array}{llcllll} (\delta\text{-MLP}) & \min & (1+\delta)\omega^T x & \geq & (\delta\text{-DLP}) & \max & \pi^T(\mathbb{1} - \Phi l) + (1+\delta)c^T l \\ & \text{s.t.} & \Phi x = \mathbb{1} & & & \text{s.t.} & \pi^T \Phi \leq (1+\delta)c^T \\ & & x \leq \mathbb{1} - l & & & & \\ & & x \geq 0, & & & & \end{array}$$

dann stellen $x_\star$ für δ-MLP und $\pi_\star$ für δ-DLP jeweils eine zulässige Lösung dar, für deren Zielfunktionswerte aber folgendes gilt:

$$\begin{aligned} \pi_\star^T(\mathbb{1} - \Phi l) + (1+\delta)\omega^T l &= \pi_\star^T(\mathbb{1} - \overline{\Phi l}) + (1+\delta)\overline{\omega}^T \overline{l} \\ &= \pi_\star^T(\mathbb{1} - \overline{\Phi l}) + \overline{\omega}^T \overline{l} + \delta\overline{\omega}^T \overline{l} \\ &= z + \delta\overline{\omega}^T \overline{l} \\ &\geq z, \end{aligned}$$

$$\begin{aligned} (1+\delta)\omega^T x_\star &= (1+\delta)\overline{\omega}^T \overline{x_\star} \\ &= (1+\delta)z. \end{aligned}$$

Aus diesen beiden Tatsachen und der Dualitätstheorie für lineare Programme folgt nun für den optimalen Zielfunktionswert [4] $z_\star$ von δ-MLP und δ-DLP folgende Abschätzung:

$$(1+\delta)z \geq z_\star \geq z.$$

Da sich aber das δ-MLP und das MLP nur durch die lineare Zielfunktionstransformation $(1+\delta)$ unterscheiden, folgt nun unmittelbar für den Optimalwert u des MLP selbige transformierte Ungleichung

$$z \geq u \geq \frac{z}{1+\delta}.$$

Schließlich führt dies zu folgenden grundlegenden Aussagen:

- z ist obere Schranke für das Optimum des MLP.
- $\frac{z}{1+\delta}$ ist untere Schranke für das Optimum des MLP, und sei fortan mit RCSP Lower Bound bezeichnet.
- $\frac{z}{1+\delta}$ ist somit auch untere Schranke für das Optimum des SPP.[5]

[4] Dessen Existenz ist trivialerweise durch die in Φ als Teilmatrix enthaltene Einheitsmatrix, die von den bereits erwähnten Tripper-Diensten induziert wird, gegeben.

[5] Dies beruht darauf, dass es sich beim MLP um eine Relaxierung des SPP bzgl. der Ganzzahligkeitsbedingungen handelt und beide Minimierungsprobleme darstellen.

3.4.1 Beobachtungen und Folgerungen

Dieser Abschnitt beschäftigt sich mit theoretischen Aussagen über die RCSP Lower Bound. Zuerst kann man sich die Frage stellen, ob diese Schranke beliebig schlecht sein kann. Das folgende Beispiel soll dies demonstrieren. Sei dazu folgender Dienstplanungsgraph mit den Dienstelementen 1 und 2 gegeben:

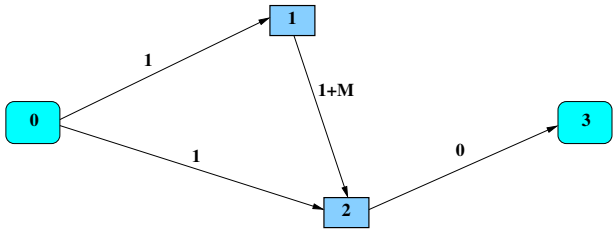

Abbildung 3.7: Dienstplanungsgraph mit 2 Elementen.

Die betrachtete Dienstmenge D enthält nur den Dienst, der beide Elemente abarbeitet. In diesem vereinfachten Szenario setzen sich die Kosten des Dienstes ausschließlich aus den Kosten der entsprechenden Bögen zusammen. Diese Annahmen führen zu folgenden linearen Programmen:

$$
\begin{array}{lll}
 & \text{(RMLP)} & \\
\min & (M+2)x_1 & \\
\text{s.t.} & \begin{pmatrix} 1 \\ 1 \end{pmatrix} x_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} & \\
 & x_1 \in [0,1] &
\end{array}
\qquad
\begin{array}{lll}
 & \text{(RDLP)} & \\
\max & \pi_1 + \pi_2 & \\
\text{s.t.} & \begin{pmatrix} \pi_1 & \pi_2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \leq M+2 & \\
 & \pi_1, \pi_2 \in \mathbb{R}. &
\end{array}
$$

Die offensichtlich optimale, weil einzige, Lösung des RMLP ist in diesem Falle $x_1 = 1$, des Weiteren sei mit $\pi_1 = 0$ und $\pi_2 = M + 2$ eine duale Optimallösung gegeben. Stellt man nun mit Hilfe dieser dualen Variablen das ACSP Modell für das PRICE Problem auf, so erhält man den Graphen mit folgenden Kosten:

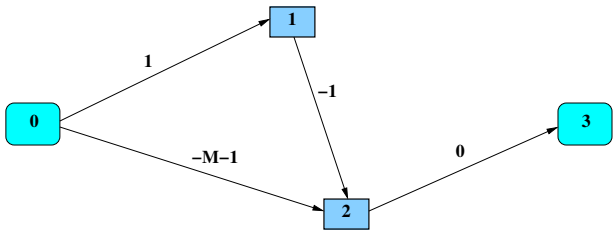

Abbildung 3.8: RCSP Relaxierung des PRICE-Problems.

Unter der Annahme, dass in diesem Beispiel jeder $(0,3)$-Pfad zu einem zulässigen Dienst korrespondiert und M eine beliebig große positive Zahl ist, findet man als Lösung des RCSP den Pfad $(0,(0,2),2,(2,3),3)$ mit negativen Kosten $-M-1$. Die Lösung des zugehörigen $\delta - PRICE$ Problemes ist $\delta = M+1$, da im folgenden modifizierten Graphen kein $(0,3)$-Pfad mit negativen Kosten existiert.

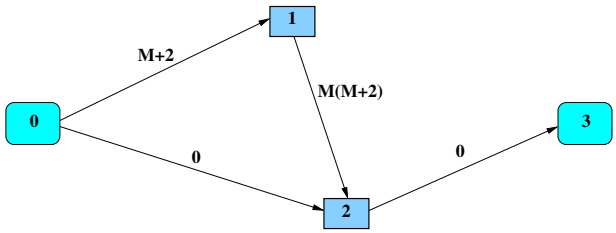

Abbildung 3.9: RCSP Relaxierung des PRICE-Problems mit δ-transformierten Kosten.

Dieses δ liefert den Wert $\frac{M+2}{1+M+1}$, also 1, als RCSP Lower Bound. Nach Hinzunahme des gefundenen Dienstes mit der Variable x_2 ergeben sich folgende linearen Programme:

	(RMLP)		(RDLP)
min	$(M+2)x_1 + x_2$	max	$\pi_1 + \pi_2$
s.t.	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$	s.t.	$\begin{pmatrix} \pi_1 & \pi_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \leq \begin{pmatrix} M+2 \\ 1 \end{pmatrix}^T$
	$x_1, x_2 \in [0,1]$		$\pi_1, \pi_2 \in \mathbb{R}$.

Um dieses Beispiel zu beenden, sei mit $\pi_1 = M + 1$ und $\pi_2 = 1$ die Optimallösung des RDLP gegeben. Durch die Nichtexistenz eines bzgl. der transformierten Kosten negativen $(0, 3)$ Pfades im betrachteten Graphen, ist die Optimalität der Lösung $x_1 = 1$ und $x_2 = 0$ gezeigt.

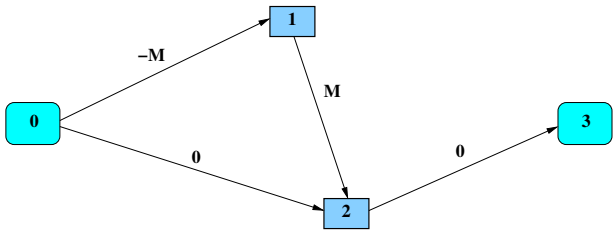

Abbildung 3.10: RCSP Relaxierung des PRICE Problemes mit transformierten Kosten.

Der Optimalwert dieses Pfadüberdeckungsproblems ist also bereits der Wert $M + 2$. Dieses Beispiel verdeutlicht damit zum einen, dass die RCSP Lower Bound beliebig schlecht sein kann, und zum anderen, dass Pfade, die zu zulässigen Diensten korrespondieren und negative reduzierte Kosten haben, nach Hinzunahme zum RMLP zu keiner echten Verbesserung des Optimalwertes führen müssen. Im Allgemeinen sind die Instanzen aus der Dienstplanung aufgrund eines hohen Symmetriegrades derart degeneriert. Erst die Hinzunahme von solchen Diensten ohne scheinbare Verbesserung kann zu 'guten' ACSP Lower Bounds führen.

Eine weitere zu diskutierende Fragestellung lautet, wie sich die Schranke zwischen den einzelnen Iterationen der Column Generation Phase verhält. Auch in diesem kleinen Beispielszenario nehmen wir an, dass sowohl alle dargestellten Pfade zulässig sind, als auch die Kosten eines Dienstes den Kosten der im Pfad enthaltenen Bögen enstprechen. Seien die, zu folgenden LPs korrespondierenden, Dienste als Startmenge und der dazugehörige Dienstplanungsgraph gegeben:

$$
\begin{array}{lll}
 & \text{(RMLP)} & \text{(RDLP)} \\
\min & \begin{pmatrix} 200 & 100 & 200 \end{pmatrix} x & \max \quad \pi^T \mathbb{1} \\
\text{s.t.} & \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} x = \mathbb{1} & \text{s.t.} \quad \pi^T \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \leq \begin{pmatrix} 200 \\ 100 \\ 200 \end{pmatrix} \\
 & x \in [0,1]^3 & \pi \in \mathbb{R}^4.
\end{array}
$$

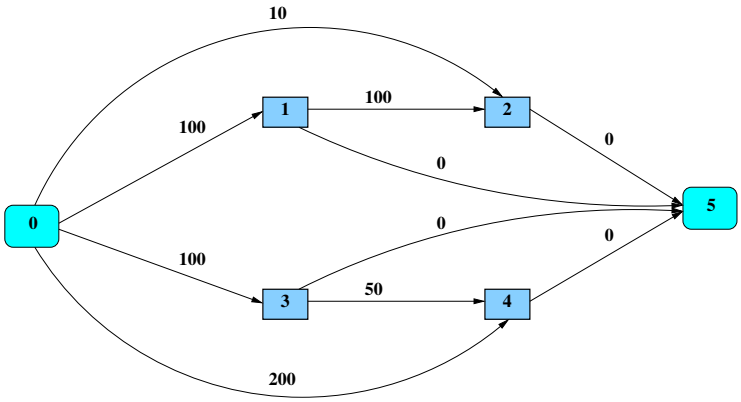

Abbildung 3.11: Dienstplanungsgraph.

Der gegenwärtige Optimalwert beider LP liegt bei 500. Sei als Lösung des RDLP $\pi_{opt} = \begin{pmatrix} 200 & 0 & 100 & 200 \end{pmatrix}^T$ gegeben, dann ergibt sich folgendes RCSP:

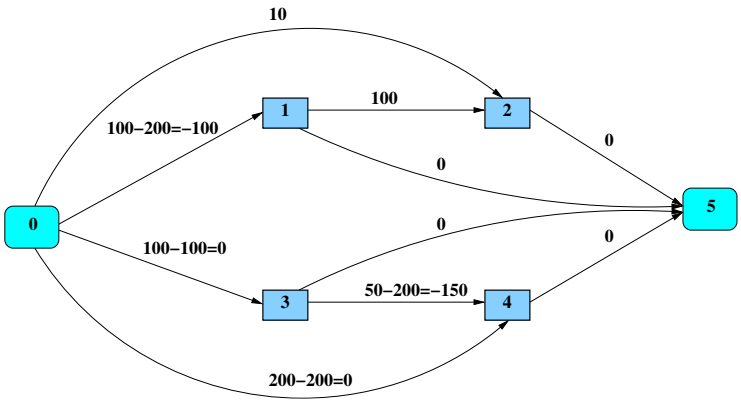

Abbildung 3.12: Dienstplanungsgraph mit transformierten Kosten.

Der kürzeste Weg von 0 nach 5 geht über die Knoten 3 und 4 und hat Kosten von -150. Wählt man $\delta = 1$, so enthält der Graph mit δ-transformierten Kosten keinen negativ bewerteten $(0, 5)$-Pfad mehr, was unmittelbar zu einer RCSP Lower Bound von 250 führt. Durch das Hinzunehmen des gefunden kürzesten Pfades ergeben sich folgende LPs:

(RMLP)

$$\min \quad \begin{pmatrix} 200 & 100 & 200 & 150 \end{pmatrix} x$$

$$\text{s.t.} \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} x = \mathbb{1}$$

$$x \in [0,1]^4$$

(RDLP)

$$\max \quad \pi^T \mathbb{1}$$

$$\text{s.t.} \quad \pi^T \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \leq \begin{pmatrix} 200 \\ 100 \\ 200 \\ 150 \end{pmatrix}$$

$$\pi \in \mathbb{R}^4$$

Der Optimalwert dieser beiden LPs liegt bei 350. Sei als Lösung des RDLP $\pi_{opt} = \begin{pmatrix} 100 & 100 & 75 & 75 \end{pmatrix}^T$ gegeben, dann betrachtet man erneut das RCSP mit transformierten Kosten.

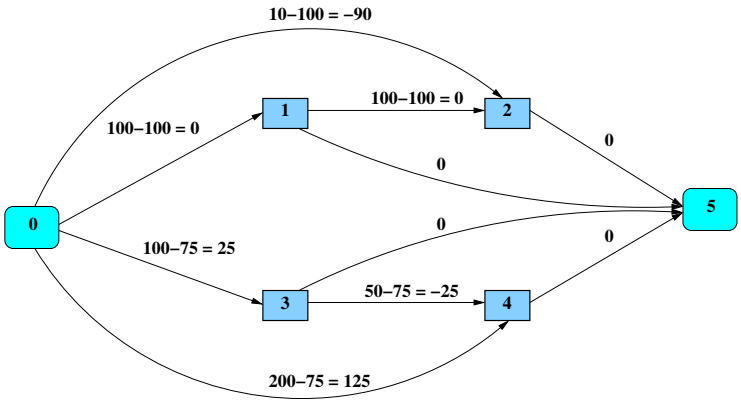

Abbildung 3.13: Dienstplanungsgraph mit transformierten Kosten.

Der kürzeste und einzige negativ bewertete Weg von 0 nach 5 führt hier ausschließlich über den Knoten 2 und hat Kosten von -90. Damit ergibt sich für δ der Wert 9 und eine RCSP Lower Bound von 35. Um das Beispiel zu beenden, sei gesagt, dass im nächsten Schritt die Optimallösung des RMLP mit 260 ermittelt und mit der Hilfe der RCSP Lower Bound als Optimallösung des zugehörigen MLP bestätigt werden kann. Dieses Beispiel liefert die Erkenntnis, dass eine neuberechnete RCSP Lower Bound keinesfalls eine bessere untere Schranke liefern muss, als eine bereits ermittelte RCSP Lower Bound.

Iteration	1	2	3
RMLP-Optimalwert	500	350	260
ACSP-Lower Bound	250	35	260

Kapitel 4

RCSP Lower Bound für DSP-Instanzen des ÖPNV

Dieses Kapitel verbindet nun die aus den letzten beiden Kapiteln gewonnenen Erkenntnisse. Zum einen wird ausführlich die RCSP Relaxierung des PRICE Problems diskutiert und deren Struktur im speziellen Szenario der Dienstplanung im ÖPNV erläutert. Zum anderen werden Möglichkeiten aufgezeigt, die dadurch gewonnenen Schranken noch zu verbessern. Dazu werden prinzipiell zwei Techniken verwendet:
Einerseits wird der Lösungsraum dekomponiert und andererseits werden weitere gültige Schnittebenen konstruiert, d.h. es wird versucht mehrere geeignete RCSP Relaxierungen, die sich nur auf eine sinnvolle Teilmenge der Dienstmenge beziehen, zu konstruieren und diese dann durch das Hinzufügen von weiteren gültigen Ungleichungen zu verschärfen. Abschließend werden ein Branch&Bound&Cut Algorithmus zum Lösen des RCSP vorgestellt und kurz die sich ergebenden Modifikationen im Falle eines dienstteilexpandierten Graphen erläutert.

4.1 RCSP Modell im ÖPNV

In diesem Abschnitt werden explizit die Vorgehensweise und die Möglichkeiten eingeführt, die zur Modellierung der RCSP-Instanzen aus den jeweiligen PRICE Problemen vorhanden und implementierbar sind. Vorgestellt werden zunächst RCSP Instanzen im ursprünglichen Dienstplanungsgraphen, d.h. die geteilten Dienste werden gemäß Abschnitt 3.1.2 modelliert. Für die Verwendung eines dienstteilexpandierten Dienstplanungsgraphen folgt im letzten Abschnitt des Kapitels eine kurze Erläuterung. Hier soll vorerst eine genauere, mit Daten aus praktischen bzw. realen Instanzen untermauerte Diskussion der eigentlichen Resource Constraints folgen.
Um den Komplex der Ressourcenbedingungen zu erläutern, benötigen wir die bereits in der Modellierung des Dienstplanungsgraphen erwähnten Bogentypen. Das liefert uns eine disjunkte Partition der Bogenmenge A, so dass gilt :

$$A = A_I \dot\cup A_{II} \dot\cup A_{III} \dot\cup A_{IV_B} \dot\cup A_{IV_E} \dot\cup A_V$$

Um die Bedingungen an die Anzahl der Dienstteile erläutern zu können, sei die Rolle der IV Bögen hier nochmals erklärt. Ein Typ IV Bogen leitet einen neuen Dienstteil ein, sofern der Zielknoten keinen Hilfsknoten darstellt, andernfalls beendet er den Dienstteil. Das liefert eine disjunkte Partition der Typ IV Bögen in Dienstteilstartbögen IV_B und Dienstteilendbögen IV_E. Der Teilpfad zwischen einem Typ IV_B Bogen und einem IV_E Bogen entspricht einem Dienstteil, sofern kein weiterer IV Bogen dazwischen liegt. Mit diesen Definitionen lassen sich folgende, an die Dienste gestellte Bedingungen modellieren.

- **Anzahl der Dienstteile**

  Maximale, minimale und optimale Anzahl der Dienstteile eines Dienstes können wie folgt als Bedingungen des Typs (3.3) und (3.4) formuliert werden. Dazu seien folgende Parameter der Dienstart bekannt :

  λ_{opt}^{Teile} - optimale Anzahl der Dienstteile
  λ_{max}^{Teile} - maximale Anzahl der Dienstteile
  λ_{min}^{Teile} - minimale Anzahl der Dienstteile

  Dann muss für den Pfad gelten:

$$\sum_{e \in A_{IV_B}} x_e + s_+^{Teile} - s_-^{Teile} = \lambda_{opt}^{Teile}$$

$$0 \leq s_+^{Teile} \leq \lambda_{opt}^{Teile} - \lambda_{min}^{Teile}$$

$$0 \leq s_-^{Teile} \leq \lambda_{max}^{Teile} - \lambda_{opt}^{Teile}$$

  Diese Bedingungen liefern bereits die gewünschten Eigenschaften. Zum einen entspricht die Anzahl der Bögen des Typs IV_B genau der Anzahl der Dienstteile, zum anderen gilt aufgrund der Nichtnegativität der Kostenkoeffizienten $d_+^{Teile} \geq 0$ und $d_-^{Teile} \geq 0$ von s_+^{Teile} und s_-^{Teile} folgendes:

$$\begin{aligned} s_+^{Teile} &\leq \lambda_{opt}^{Teile} - \lambda_{min}^{Teile} \\ s_+^{Teile} &\leq \sum_{e \in A_{IV_B}} x_e + s_+^{Teile} - s_-^{Teile} - \lambda_{min}^{Teile} \\ \lambda_{min}^{Teile} &\leq \sum_{e \in A_{IV_B}} x_e - s_-^{Teile} \leq \sum_{e \in A_{IV_B}} x_e. \end{aligned}$$

  Analog gilt:

$$\begin{aligned} s_-^{Teile} &\leq \lambda_{max}^{Teile} - \lambda_{opt}^{Teile} \\ s_-^{Teile} &\leq \lambda_{max}^{Teile} - (\sum_{e \in A_{IV_B}} x_e + s_+^{Teile} - s_-^{Teile}) \\ 0 &\leq \lambda_{max}^{Teile} - \sum_{e \in A_{IV_B}} x_e - s_+^{Teile} \\ \sum_{e \in A_{IV_B}} x_e &\leq \lambda_{max}^{Teile} - s_+^{Teile} \leq \lambda_{max}^{Teile}. \end{aligned}$$

- **Anzahl der Dienststücke**

Maximale, minimale und optimale Anzahl der Dienststücke eines Dienstes können analog als Bedingungen des Typs (3.3) und (3.4) formuliert werden. Dazu seien folgende Parameter der Dienstart bekannt :

$\lambda_{opt}^{Stück}$ - optimale Anzahl der Dienststücke
$\lambda_{max}^{Stück}$ - maximale Anzahl der Dienststücke
$\lambda_{min}^{Stück}$ - minimale Anzahl der Dienststücke

Da bei Beginn bzw. Wechsel des Dienstteiles auch ein neues Dienststück beginnt, muss gemäß analoger Argumentation gelten:

$$\sum_{e \in A_{III} \cup A_{IV_B}} x_e + s_+^{Stück} - s_-^{Stück} = \lambda_{opt}^{Stück}$$

$$0 \leq s_+^{Stück} \leq \lambda_{opt}^{Stück} - \lambda_{min}^{Stück}$$
$$0 \leq s_-^{Stück} \leq \lambda_{max}^{Stück} - \lambda_{opt}^{Stück}$$

- **Schichtzeit**

Als Schichtdauer bzw. -zeit wird die gesamte Zeitspanne von Beginn des Dienstes bis zum Ende des Dienstes bezeichnet, inklusive der bei geteilten Diensten auftretenden nicht als Pause anrechenbaren Dienstunterbrechungen. Auch hier existieren in der Praxis strikte Vorgaben in Bezug auf die maximal erlaubte Schichtdauer und eine Mindestlänge, die es einzuhalten gilt.
Im eingeführten Dienstplanungsgraphen wird das mit Hilfe folgender Daten modelliert:

$st(v)$ - Startzeit des Knotens $v \in V_D \cup V_E$
$et(v)$ - Endzeit des Knotens $v \in V_D \cup V_E$

$\lambda_{opt}^{Schichtzeit}$ - optimale Schichtzeit des kompletten Dienstes
$\lambda_{max}^{Schichtzeit}$ - maximale Schichtzeit des kompletten Dienstes
$\lambda_{min}^{Schichtzeit}$ - minimale Schichtzeit des kompletten Dienstes

Für jeden Bogen $e = (u, v) \in A$ wird nun typspezifisch die Ressource r_e, $r \in \mathbb{R}_+^{|A|}$ für die Schichtdauer des Dienstes definiert, wobei die Idee, die Parameter des Zielknotens v auf den Bogen e zu transformieren, umgesetzt wird.

$$r_{(u,v)} = \begin{cases} et(v) - et(u) & , (u,v) \in A_I \cup A_{II} \cup A_{III} \cup A_V \\ et(v) - st(v) & , (u,v) \in A_{IV_B} \\ 0 & , (u,v) \in A_{IV_E} \end{cases}$$

Dadurch lässt sich die Schichtdauer mit gegebenen Dienstartparametern folgendermaßen als Ressource Constraint formulieren:

$$r^T x + s_+^{Schichtzeit} - s_-^{Schichtzeit} = \lambda_{opt}^{Schichtzeit}$$

$$0 \leq s_+^{Schichtzeit} \leq \lambda_{opt}^{Schichtzeit} - \lambda_{min}^{Schichtzeit}$$
$$0 \leq s_-^{Schichtzeit} \leq \lambda_{max}^{Schichtzeit} - \lambda_{opt}^{Schichtzeit}$$

- **Dienstzeit**

  Als Dienstzeit wird die gesamte Zeitspanne von Beginn des Dienstes bis zum Ende des Dienstes bezeichnet, exklusive der bei geteilten Diensten auftretenden nicht als Pause anrechenbaren Dienstunterbrechungen. Daraus folgt unmittelbar, dass bei zusammenhängenden Diensten diese Bedingungen äquivalent zu denen der Schichtdauer und somit redundant sind. Andernfalls ergibt sich mit gegebenen Parametern und nach analoger Vorgehensweise folgende Resource Constraint für die Dienstdauer :

$$r_{(u,v)} = \begin{cases} et(v) - et(u) & , (u,v) \in A_I \cup A_{II} \cup A_{III} \\ et(v) - st(v) & , (u,v) \in A_{IV_B} \\ 0 & , (u,v) \in A_{IV_E} \cup A_V \end{cases}$$

$\lambda_{opt}^{Dienstzeit}$ - optimale Dienstzeit des kompletten Dienstes
$\lambda_{max}^{Dienstzeit}$ - maximale Dienstzeit des kompletten Dienstes
$\lambda_{min}^{Dienstzeit}$ - minimale Dienstzeit des kompletten Dienstes

$$r^T x + s_+^{Dienstzeit} - s_-^{Dienstzeit} = \lambda_{opt}^{Dienstzeit}$$
$$0 \leq s_+^{Dienstzeit} \leq \lambda_{opt}^{Dienstzeit} - \lambda_{min}^{Dienstzeit}$$
$$0 \leq s_-^{Dienstzeit} \leq \lambda_{max}^{Dienstzeit} - \lambda_{opt}^{Dienstzeit}$$

- **Gesamtlenkzeit**

  In der Lenkzeitverordnung existieren Restriktionen für die Gesamtlenkzeit, die ebenfalls als minimale, optimale und maximale Bedingungen gefasst werden können. Hierbei sei für jeden Knoten $v \in V_D \cup V_E$ mit lz_v die Lenkzeit des Dienstelementes bezeichnet, die trivialerweise für $v \in V_H$ Null gesetzt wird. Für jeden Bogen $e = (u,v) \in A$ entspricht damit die Ressource r_e, $r \in \mathbb{R}_+^{|A|}$, die die Gesamtlenkzeit modelliert, lz_v.

$\lambda_{opt}^{Lenkzeit}$ - optimale Gesamtlenkzeit des kompletten Dienstes
$\lambda_{max}^{Lenkzeit}$ - maximale Gesamtlenkzeit des kompletten Dienstes
$\lambda_{min}^{Lenkzeit}$ - minimale Gesamtlenkzeit des kompletten Dienstes

$$r^T x + s_+^{Lenkzeit} - s_-^{Lenkzeit} = \lambda_{opt}^{Lenkzeit}$$
$$0 \leq s_+^{Lenkzeit} \leq \lambda_{opt}^{Lenkzeit} - \lambda_{min}^{Lenkzeit}$$
$$0 \leq s_-^{Lenkzeit} \leq \lambda_{max}^{Lenkzeit} - \lambda_{opt}^{Lenkzeit}$$

Im Allgemeinen sind dies die Eigenschaften, die alle Instanzen aus der ÖPNV Dienstplanung gemein haben. Im Speziellen kann natürlich dieser Komplex erweitert werden, sofern es sich um lineare und eindeutig den Bögen zurechenbare Sachverhalte handelt. Arbeitszeit, Pausenzeit oder bezahlte Zeit sind jedoch so meist nicht sinnvoll zu modellieren, da eben diese Zuordnung nicht getroffen werden kann.
Das Problem ist hierbei vor allem, dass zwar die mögliche Pausenzeit eines Knotens bzw. die Unterbrechungszeit eines Bogens feststehen, ob sie jedoch wirklich als Pause anrechenbar sind,

hängt von der gesamten Struktur des Dienstteiles ab.
Somit kann man, da sowohl die Arbeitszeit als auch die bezahlte Zeit eines Dienstes abhängig von der angerechneten Pausenzeit des Dienstes sind, jene Restriktionen vorerst nicht erfüllen. Eine mögliche Vorgehensweise wird im nächsten Abschnitt vorgestellt, dazu muss jedoch das Pausenregelwerk explizit eingeführt werden.

4.2 Relaxierungen der Pausenregelungen

Man kann die Menge der zulässigen Dienste nach Dienstarten unterteilen, wie in Abbildung 4.1 am Beispiel dargestellt, und so für jede Dienstart ein RCSP, als Relaxierung des PRICE Problemes für diese Dienstart, aufstellen (siehe Abbildung 4.2) . Diese Relaxierungen unterscheiden sich vor allem durch die Resource Constraints, in denen sich die dienstartspezifischen Parameter, die im letzten Abschnitt erläutert worden, wiederspiegeln. Aber auch die Bogenmenge des zugrundeliegenden Dienstplanungsgraphen kann von Dienstart zu Dienstart abweichen, da die Definition der Verknüpfungen ebenfalls dienstartspezifisch ist.

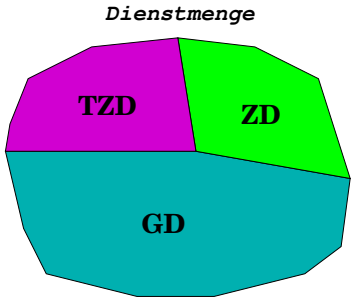

Abbildung 4.1: Partition der zulässigen Pfad/Dienstmenge in geteilte, zusammenhängende und Teilzeitdienste.

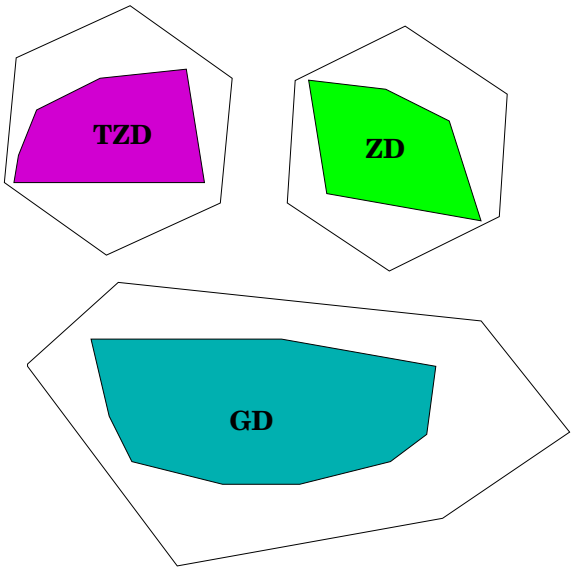

Abbildung 4.2: RCSP Relaxierungen der einzelnen Dienstarten.

Berechnet man nun für jede Dienstart gemäß Kapitel 3 ein δ und eine dazugehörige RCSP Lower Bound, so liefert das Maximum aller eine globale, d.h. für die gesamte Dienstmenge gültige, RCSP Lower Bound für das MLP. Um aber die untere Schranke jeder einzelnen Dienstart-Relaxierung zu verbessern, hilft folgende Idee weiter:
Die Menge der Dienste einer Dienstart wird weiter unterteilt bzw. überdeckt durch andere sinnvoll charakterisierbare Mengen, wie in Abbildung 4.3 demonstriert. Dieses Vorgehen wird anhand der Pausenregelungen in diesem Abschnitt vorgestellt.
Angenommen man hat die Menge $\mathcal{D}$ der zulässigen Dienste mit Kostenvektor $\omega \in \mathbb{R}^{|\mathcal{D}|}$ zu einer gegebenen Dienstart mit den Mengen M_i für $i = 0, 1, \ldots, k$ komplett überdeckt, dann folgt trivialerweise, dass das Minimum über alle Elemente dieser Mengen kleiner oder gleich dem Minimum für die spezielle Dienstart ist. Damit stellt

$$\min_{x \in \bigcup_{i=0}^{k} M_i} \omega^T x$$

eine Relaxierung des eigentlichen Problems

$$\min_{x \in D} \omega^T x$$

dar.

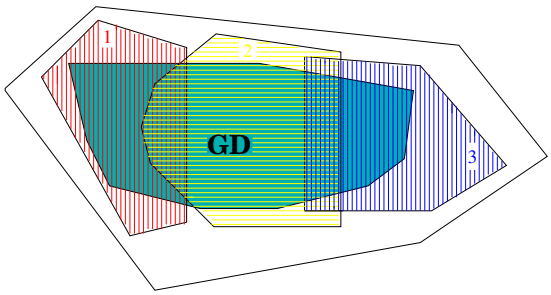

Abbildung 4.3: Eine nicht disjunkte Überdeckungen der RCSP Relaxierung für geteilte Dienste bzgl. dreier unterschiedlicher Pausenregelungrelaxierungen.

Es wird nun deutlich, warum wir diese Modellierung und das Lösen mit Hilfe von IP-Solvern gewählt haben. Auf der einen Seite ist es der Vorteil der schnellen Modifikation der Menge der Resource Constraints, auf der anderen die Möglichkeit, nicht lineare komplexe Zusammenhänge durch lineare Bedingungen zu relaxieren.
Innerhalb von Dienstteilen wird durch gesetzliche und tarifliche Vorgaben festgelegt, inwieweit Dienstunterbrechungen als Pause im Sinne des Arbeitzeitgesetzes und der Lenkzeitverordnung anerkannt werden. Dazu seien folgende auftretende Arten von Pausenregelungen spezifiziert:

- **Keine Pausenanrechnung** bedeutet, dass es zu keiner Anrechnung von Unterbrechungen als Pause kommt, somit also beispielswiese nur die maximale Arbeits- bzw. Lenkzeit pro Dienstteil eingehalten werden muss.

- **Blockpausen mit fester Anzahl von notwendigen Blöcken** sind zusammenhängende Pausen, die in Anzahl der Blöcke und Mindestdauer je Block rechtlich vorgeschrieben zu gewähren sind. Außerdem existieren sogenannte minimale und maximale Arbeistzeitfenster vor und nach dem jeweiligen angerechneten Pausenblock.

- **Quotientenregelungen** sind Bedingungen, die das Verhältnis zwischen Arbeits- bzw. Lenkzeit und der Summe aller angerechneten Pausen regeln. In der Praxis spielen hier 1/5 und 1/6 Regeln pro Dienstteil eine Rolle. Auch für diese Regeln existieren sog. minimale und maximale Arbeistzeitfenster vor und nach dem jeweiligen angerechneten Pausenblock.

In der Dienstplanung des ÖPNV sind folgende Anrechnungsarten einer Dienstunterbrechung möglich:

- **Notwendig bzw. Obligatorisch**, d.h. die Dienstunterbrechung wird als notwendige Pause betrachtet und angerechnet.

- **Optional**, d.h. die Unterbrechung wird als nicht notwendige Pause betrachtet.

- **Nicht angerechnet**, d.h. die Dienstunterbrechung wird, wenn auch möglicherweise anrechenbar, nicht als Pause bewertet.

Die Relevanz dieser Anrechnungsarten sei kurz anhand der Ein-Blockpausenregelung für Dienste, die aus einem Dienstteil bestehen, erläutert. Würde man beispielsweise gleich zu Beginn des Dienstes eine Dienstunterbrechung als Pause anrechnen, so wäre das Arbeitszeitfenster vor der ersten Blockpause schlecht ausgenutzt, und das folgende Arbeitszeitfenster würde die gesamte Dienstteildauer und somit die gesamte Dienstlänge beschränken. In diesem Fall wäre es also sinnvoll, diese Unterbrechung als optional oder nichtangerechnet zu betrachten. Der Unterschied zwischen diesen beiden besteht darin, dass bei optionaler Anrechnung die ununterbrochene Arbeitszeit zwischen zwei Pausen auf Null gesetzt wird, während sie bei Nicht Anrechnung fortgeschrieben wird.
Mit Hilfe dieses Regelwerkes kann nun die Optimierung den Bedürfnissen entsprechend Dienste bilden und abschließend die Kombination von Pausenregel und Anrechnungsart zielgerichtet wählen.
Nach dieser Einführung des Pausenregelwerkes sollen nun Möglichkeiten vorgstellt werden, diese Restriktionen durch relaxierte Resource Constraints in das Modell zu integrieren.
Das Problem liegt darin, dass man a priori nicht weiß, ob die Dienstunterbrechung als Pause angerechnet wird oder nicht. Deshalb kann man, da die Arbeitszeit von der gewährten Pausenzeit abhängt, auch keine für jeden Dienst exakte Zuordnung zwischen Bogen bzw. Knoten und tatsächlicher Arbeitszeit treffen. Dies löst man mit Hilfe von bekannten zulässigen Intervallen. Sei dazu für jeden Bogen $(i, j) \in A$ des Dienstplanungsgraphen folgende Information abhängig von der jeweiligen Anrechnungsart gegeben:

- minimale anrechenbare Pausenzeit $(b_{min})_{(i,j)} \in \mathbb{N}$
- maximale anrechenbare Pausenzeit $(b_{max})_{(i,j)} \in \mathbb{N}$
- minimale anrechenbare Arbeitszeit $(w_{min})_{(i,j)} \in \mathbb{N}$
- maximale anrechenbare Arbeitszeit $(w_{max})_{(i,j)} \in \mathbb{N}$.

Dann gilt für die tatsächliche Pausenzeit eines Bogens $0 \leq b_{(i,j)} \leq (b_{max})_{(i,j)}$ und für die tatsächliche Arbeitszeit eines Bogens $(w_{min})_{(i,j)} \leq w_{(i,j)} \leq (w_{max})_{(i,j)}$.
Welche Werte für die zulässigen Intervalle gewählt werden ist problemspezifisch zu ermitteln. Für die betrachteten Instanzen wurde als maximale Arbeitszeit eines Bogens die Dauer und als minimale Arbeitszeit die Dauer abzüglich der maximalen anrechenbaren Pausenzeit gewählt. Die Werte des Elementes j werden dabei auf den Bogen (i, j) transformiert, wie bei anderen Parametern bereits vorgestellt, .
Ob das zugrundeliegende Merkmal dies zulässt, muss jedoch speziell geprüft werden. Da sich aber das Pausenregelwerk auf Dienstteile und nicht komplette Dienste bezieht, d.h. insbesondere die Kombination von verschiedenen Pausenregelungen in einem geteilten Dienst ist zulässig, muss hier das Problem weiter differenziert werden.
Dies führt, wie der folgende Abschnitt zeigen wird, zu unterschiedlichen RCSP Relaxierungen, die sich in den dienstartspezifischen Resource Constraints nicht unterscheiden. Man beachte hierbei, dass die oben vorgestellten Parameter sowohl von der Pausenregelung als auch von der Anrechnungsart abhängen können. Sei für die folgenden Betrachtungen das dienstartspezifische RCSP Modell mit $x \in \{0,1\}^{|A|}$ gegeben:

4.2.1 Dienstarten mit genau einem Dienstteil

Keine Pausenanrechnung

In diesem Fall gelten die Beschränkungen der Arbeitszeit pro Dienstteil W_{max} und W_{min} trivialerweise für den gesamten aus einem Dienstteil bestehenden Dienst.
Bögen, deren minimale anrechenbare Pausenzeit echt positiv ist oder für die die Anrechnungsart drei nicht in Frage kommt, können für die Relaxierung dieser Pausenregelung aus dem Digraphen entfernt werden, da sie offenbar in keinem zulässigen Dienst dieser Pausenregelung enthalten seien können.
Damit die Bedingung bzgl. der maximalen Arbeitszeit eingehalten wird, darf zumindestens die Summe aus den minimalen Arbeitszeiten der jeweiligen Elemente diese nicht überschreiten. Analog muss die Summe der maximal anrechenbaren Arbeitszeit der Elemente mindestens der minimalen Arbeitszeit entsprechen. Diese beiden Ungleichungen stellen eine mögliche Relaxierung dieser Pausenregelung dar und können somit der RCSP Relaxierung für diese Pausenregelung hinzugefügt werden. Dabei sei mit $w \in \mathbb{N}^{|A|}$ die tatsächliche Arbeitszeit auf den Bögen des Dienstes bezeichnet.

$$w_{min}^T x \leq w^T x \leq W_{max}$$

$$w_{max}^T x \geq w^T x \geq W_{min}$$

Blockpausen mit fester Anzahl von notwendigen Blöcken

Auch für diese Pausenregelung existieren Beschränkungen der Arbeitszeit pro Dienstteil W_{max} und W_{min}. Des Weiteren soll die Summe der maximal als Pause anrechenbaren Dienstunterbrechungen mindestens der für die gewählte Blockpause gegebenen Mindestdauer $T \in \mathbb{N}$ entsprechen.
Außerdem kann man noch die Anzahl der Blöcke modellieren, indem man jedem Bogen mit $(b_{max})_{(i,j)} > T$ einen neuen zusätzlichen Hilfsressourcenwert $(\overline{b}_{max})_{(i,j)} = 1$ und andernfalls 0 zuweist. Über den gesamten Pfad summiert, muss der Wert dieser Ressource mindestens $B \in \mathbb{N}$, die Anzahl der zu gewährenden Pausenblöcke, ergeben.
Seien mit $b \in \mathbb{N}^{|A|}$ die tatsächlichen Pausendauern auf den Bögen des Dienstes bezeichnet, mit $w \in \mathbb{N}^{|A|}$ die tatsächliche Arbeitszeit und mit $\overline{b} \in \{0,1\}^{|A|}$ der dazugehörige Hilfsressourcenvektor, so gilt:

$$w_{min}^T x \leq w^T x \leq W_{max}$$

$$w_{max}^T x \geq w^T x \geq W_{min}$$

$$b_{max}^T x \geq b^T x \geq BT$$

$$\overline{b}_{max}^T x \geq \overline{b}^T \mathbb{1} = B.$$

Quotientenregelung

Für diese Pausenregelung existieren ebenfalls Beschränkungen der Arbeitszeit pro Dienstteil W_{max} und W_{min}. Sei außerdem q der entsprechende Quotient, der das Verhältnis zwischen Pausenzeit und Arbeitszeit angibt. So muss zumindestens die maximal anrechenbare Pausenzeit zu der minimal anrechenbaren Arbeitszeit diesem Verhältnis entsprechen. Das führt zu folgenden Ungleichungen, wobei wiederum die tatsächlichen Arbeitszeiten $w \in \mathbb{N}^{|A|}$ und Pausendauern $b \in \mathbb{N}^{|A|}$ a priori nicht bekannt sind:

$$w_{min}^T x \leq w^T x \leq W_{max}$$

$$w_{max}^T x \geq w^T x \geq W_{min}$$

$$b_{max}^T x \geq b^T x \geq \tfrac{1}{q} w^T x \geq \tfrac{1}{q} w_{min}^T x.$$

4.2.2 Dienstarten mit mehreren Dienstteilen

Da es erlaubt ist, in verschiedenen Dienstteilen verschiedene Pausenregelungen anzuwenden, stellt es sich hier als erheblich schwieriger heraus sinnvolle Relaxierungen im ursprünglichen Dienstplanungsgraphen zu konstruieren.
Wählt man eine Relaxierung, die ausschließlich Aussagen über den ersten Dienstteil macht, so wird aufgrund der erheblichen Freiheitsgrade für die nachfolgenden Dienstteile vermutlich keine gute Schranke ermittelt werden können. Wir wollen nun die Vorgehensweise für geteilte Dienste exemplarisch für zweigeteilte Dienste durchführen.

Keine Pausenanrechnung in beiden Dienstteilen

Seien wiederum Beschränkungen der Arbeitszeit pro Dienstteil W_{max} und W_{min} bekannt. Des Weiteren sei mit $w_1, w_2 \in \mathbb{N}$ die tatsächliche Arbeitszeit der jeweiligen Dienstteile bezeichnet, so folgt nach analoger Argumentation:

$$\begin{aligned} w_{min}^T x &\leq w_1 + w_2 \leq 2\,W_{max}^{dt} \\ w_{max}^T x &\geq w_1 + w_2 \geq 2\,W_{min}^{dt}. \end{aligned}$$

Schon für die einfachste Pausenregelung fällt bei dieser Art der Relaxierung auf, dass sie für mehrere Dienstteile an Schärfe verliert. Zwar modelliert diese relaxierte Ungleichung, dass die gewünschte Eigenschaft für den gesamten Dienst (Pfad) gilt, ob sie jedoch auch für beide Dienstteile (Teilpfade) einzeln betrachtet erfüllt ist, kann nicht sichergestellt werden. Der Vollständigkeit halber werden trotzdem weitere mögliche Relaxierungen der Kombination von Pausenregelungen betrachtet, um letzendlich die Konstruktion des dienstteilexpandierten Graphen zu begründen und zu motivieren.

Blockpausenregelung in beiden Dienstteilen

Seien dazu mit T_1 die Mindestdauer der Pausenblöcke im ersten Dienstteil, mit T_2 die des zweiten und mit B_1 und B_2 deren Anzahl gegeben. Seien weiterhin mit $b_1, b_2 \in \mathbb{N}$ die tatsächliche Anzahl angerechneter Pausen in Diensteil 1 bzw. 2 und mit $t_1, t_2 \in \mathbb{N}^{|A|}$ die tatsächlichen Pausendauern bezeichnet. Dann folgt mit der bereits eingeführten Definition von $\overline{b}_{max} \in \{0,1\}^{|A|}$ und analoger Argumentation:

$$\begin{aligned} b_{max}^T x &\geq t_1^T x + t_2^T x \geq B_1 T_1 + B_2 T_2 \\ \overline{b}_{max}^T x &\geq b_1 + b_2 = B_1 + B_2. \end{aligned}$$

Quotientenregelung in beiden Dienstteilen

Seien mit $w \in \mathbb{N}$ die tatsächliche Arbeitszeit, mit $b \in \mathbb{N}^{|A|}$ die tatsächlich angerechneten Pausendauern bezeichnet und mit q der zu erfüllende Quotient gegeben, dann führt dies zu folgender Ungleichung:

$$b_{max}^T x \geq b^T x \quad \geq \quad \frac{1}{q} w^T x \geq \frac{1}{q} w_{min}^T x.$$

Mindestens ein Dienstteil mit Blockpausenregelung

Hier wird der Fall betrachtet, dass in einem Dienstteil die Blockpausenregel zur Anwendung kommt. Seien wiederum $b \in \mathbb{N}^{|A|}$ und $\overline{b} \in \mathbb{N}^{|A|}$, dann stellt für die gegebene Mindestdauer $T \in \mathbb{N}$ der Blöcke, für die Anzahl $B \in \mathbb{N}$ der Blöcke und für die Vektoren $\overline{b}_{max} \in \{0,1\}^{|A|}$ und $b_{max} \in \mathbb{N}^{|A|}$ die folgende Ungleichung eine gültige Relaxierung dar:

$$\begin{aligned} b_{max}^T x &\geq b^T x \geq BT \\ \overline{b}_{max}^T x &\geq \overline{b}^T x = B. \end{aligned}$$

Man beachte jedoch, dass alle zulässigen Lösungen der Blockpausenregelung in beiden Dienstteilen auch für diese Relaxierung immer zulässig sind. Daraus folgt bereits, dass die untere Schranke, die die hier betrachtete Relaxierung liefert, immer schlechter sein wird als die Schranke der Relaxierung, für die die Blockpausenregelung in beiden Dienstteilen gefordert wird.

Kombination von Quotientenregelung und keine Pausenanrechnung

Sei W_{max} die maximale Arbeitszeit für die Pausenregelung keine Pausenanrechnung. Des Weiteren seien mit $w \in \mathbb{N}$ die tatsächliche Arbeitszeiten und mit $b \in \mathbb{N}$ die angerechneten Pausendauern des Dienstteiles bezeichnet, der die Quotientenregelung erfüllt. Dann muss, da für den anderen Dienstteil die maximale Arbeitszeitbedingung erfüllt sein muss, Folgendes gelten :

$$b_{max}^T x \geq b \quad \geq \quad \frac{w}{q} \geq \frac{1}{q}(w_{min}^T x - W_{max}^{dt})$$

Motivation des dienstteilexpandierten Dienstplanungsgraphen

Die Problematik, für geteilte Dienste sinnvolle Relaxierungen zu finden und das dafür verantwortliche Problem der Dienstteilzuordnung führten letztendlich zur Idee, den bereits vorgestellten dienstteilexpandierten Dienstplanungsgraphen zu konstruieren. Dadurch ist man in der Lage, die Menge der Dienste einer Dienstart, die aufgrund der Pausenregelungen in sich

heterogen ist, in unterschiedliche, in sich homogene Teilmengen aufzuteilen.
Insgesamt stellt dieser Abschnitt die Idee vor, mehrere, kleinere und stärkere Relaxierungen zu lösen, anstatt einer Relaxierung, die die Pausenregelung komplett vernachlässigt. Ein Problem, das hierbei zum Vorschein kommt, ist, dass je mehr Dienstarten und Pausenregelungen existieren, desto mehr Relaxierungen aufgestellt und gelöst werden müssen.

4.3 Schnittebenen

Das Ziel ist es, die RCSP Relaxierung des PRICE Problemes zu verschärfen und weitere Diensteigenschaften in das Modell zu integrieren. Die offensichtlichen linearen Bedingungen wurden im ersten Abschnitt ausführlich vorgestellt. In diesem Abschnitt sollen Eigenschaften, die sich auf die Dienstuntereinheiten, d.h Teile des Pfades, beziehen, diskutiert werden.
Auf der einen Seite wird versucht, Pfade, die diese Eigenschaften nicht besitzen, durch geeignete Schnittebenen zu verbieten, auf der anderen kann man durch weitere Bedingungen an die Pfade gewisse Pfadeigenschaften erzwingen. Zwei Klassen von Ungleichungen sind dabei von Bedeutung:

- Zum einen die Klasse der Infeasible Set Constraints (ISC). Durch diese werden Lösungen, die unzulässige Mengen $S \subset A$ enthalten, d.h. die nicht gemeinsam in einer zulässigen Lösung auftreten können, abgeschnitten.

$$\sum_{e \in S} x_e \quad \leq \quad |S| - 1 \quad \text{(ISC)}$$

- Zum anderen die Klasse der Feasible Set Constraints (FSC). Durch diese werden Bedingungen folgender Art modelliert:
  Zu einem Bogen $a \in A$ kann man eine Menge $T \subset A$ finden, so dass jede zulässige Lösung, die a enthält, auch mindestens n Elemente der Menge T enthalten muss.

$$x_a \quad \leq \quad \frac{1}{n} \sum_{e \in T} x_e \quad \text{(FSC)}$$

In den folgenden Abschnitten wird versucht Diensteigenschaften mit Hilfe dieser Klassen von Ungleichungen zu modellieren.

4.3.1 Verbesserung der Schranke durch PDCs

Dienstteildauer

Durch die Dienstteil-Ressourcenbedingung wird, wie im letzten Abschnitt erläutert, die zulässige Anzahl der Dienstteile modelliert. Es existieren jedoch auch Vorgaben für die Dauer solcher Dienstteile. Dazu seien folgende Parameter bekannt:

$st(e)$	- Startzeit des Bogens $e = (u, v)$, entspricht der Startzeit von u
$et(e)$	- Endzeit des Bogens $e = (u, v)$, entspricht der Endzeit von v
$\lambda_{min}^{Dientteildauer}$	- minimal zulässige Dauer der einzelnen Dienstteile
$\lambda_{max}^{Dientteildauer}$	- maximal zulässige Dauer der einzelnen Dienstteile.

Dadurch lassen sich für jeden Bogen $a \in A_{IV_B}$ folgende zulässigen Bogenmengen definieren:

$$\Pi_a = \{b \in A_{IV_E} \,|\, \lambda_{min}^{Dientteildauer} \leq et(b) - st(a) \leq \lambda_{max}^{Dientteildauer}\}$$

Dementsprechend lässt sich für jeden Bogen $a \in A_{IV_B}$ folgende Menge unzulässiger Typ IV_E Bögen finden:

$$\overline{\Pi}_a = \{b \in A_{IV_E} \,|\, 0 < et(b) - st(a) < \lambda_{min}^{Dientteildauer}\}$$

Die folgende Klasse von Ungleichungen gewährleistet somit, dass für jeden Typ IV_B Bogen im Pfad auch mindestens ein zulässiger Typ IV_E Bogen enthalten ist, der das Ende des Dienstteiles symbolisiert (PDC_I).
Andererseits darf auf dem Pfad zu keinem enthaltenen Typ IV_B Bogen ein, gemäß dem oben definierten Unzulässigkeitsbegriff für diesen Bogen unzulässiger, Typ IV_E Bogen auf dem Pfad liegen (PDC_II).

$$\begin{aligned} x_a &\leq \sum_{b \in \Pi_a} x_b & \forall\, a \in A_{IV_B} \quad (\text{PDC}_\text{I}) \\ x_a + \sum_{b \in \overline{\Pi}_a} x_b &\leq 1 & \forall\, a \in A_{IV_B} \quad (\text{PDC}_\text{II}) \end{aligned}$$

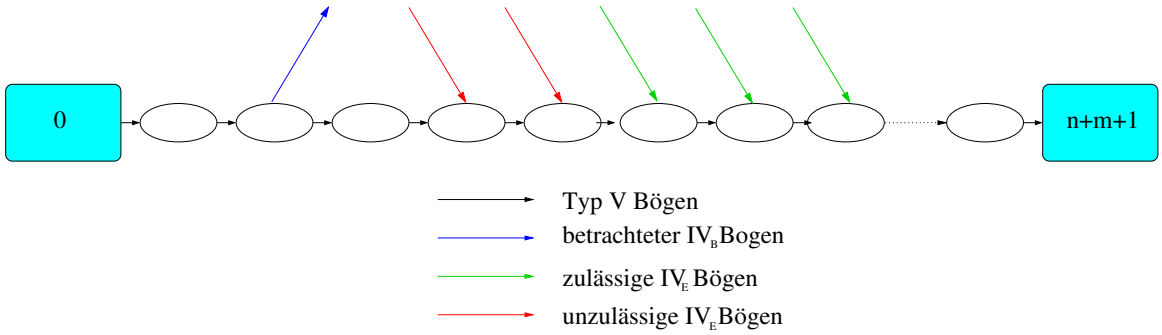

Abbildung 4.4: Zulässigkeitsbegriff für Typ IV_B Bögen.

Diese Bedingungen wollen wir allgemein als Part/Piece Duration Constraints, kurz PDCs, bezeichnen. Dass diese Bezeichnung gerechtfertigt ist, zeigt der folgende Beweis:

BEHAUPTUNG:

Sei x_{opt} Lösung des RCSP Modells mit PDCs und P der durch x_{opt} induzierte Lösungspfad. Dann gelten für alle induzierten Dienstteile die Bedingungen bzgl. deren Mindest- und Maximaldauer.

BEWEIS:

Angenommen es existiert nun in P ein Dienstteil, dessen Gesamtdauer die Mindestdauer unter- oder die Maximaldauer überschreitet. Sei $a \in A_{IV_B}$ der Bogen, der den Dienstteil einleitet, und $b \in A_{IV_E}$ der Bogen, der das Ende dieses Dienstteiles ist, dann existiert per Konstruktion des Dienstteiles kein weiterer Typ IV Bogen in diesem Teilpfad.

- 1. Fall:
$$et(b) - st(a) > \lambda_{max}^{Dientteildauer},$$
dann gilt offenbar $b \notin \Pi_a$. Andererseits gilt für jeden Bogen $e \in \Pi_a$ $et(e) < et(b)$ und $st(e) > st(a)$, d.h. jeder für a zulässige Typ IV_E Bogen beginnt zeitlich nach a und endet vor b. Da nun aber trivialerweise die Startzeiten und Endzeiten der Bögen im Pfad jeweils aufsteigend sind, kann also kein Bogen $e \in \Pi_a$ mehr im Restpfad enthalten sein, d.h. weder vor noch nach dem betrachteten Dienstteil. Somit ist gezeigt, dass die Bedingung
$$x_a \leq \sum_{e \in \Pi_a} x_e \ \ (\text{PDC}_\text{I})$$
für x_{opt} verletzt sein muss.

- 2. Fall:
$$et(b) - st(a) < \lambda_{min}^{Dientteildauer}$$
Offenbar gilt nicht nur $b \notin \Pi_a$, sondern auch $b \in \overline{\Pi}_a$. Somit ist in diesem Fall, da a und b im Pfad enthalten sind, die Ungleichung bzgl. der unzulässigen Typ IV_E Bögen von a verletzt:
$$x_a + \sum_{e \in \overline{\Pi}_a} x_e \leq 1 \ \ (\text{PDC}_\text{II}).$$

Da wir die Annahme in beiden Fällen zum Widerspruch führen konnten, ist gezeigt, dass für alle induzierten Dienstteile die Bedingungen bzgl. deren Mindest- und Maximaldauer durch die PDCs erfüllt sind.
Andererseits erfüllt jeder zulässige, zu einem Dienst korrespondierende, Pfad $P \in \mathcal{FP}$ die Bedingungen PDC_I und PDC_II. Für die PDC_I ist das offensichtlich. Für die PDC_II jedoch bleibt zu argumentieren, dass zwei Bögen aus derselben Menge Π_a nicht in einem zulässigen Pfad enthalten sein können. Aber auch das liegt auf der Hand, wenn man bedenkt, dass diese Bögen zeitlich zu früh enden, um mit dem Bogen a in einem zulässigen Dienstteil zu sein. Dann kann aber auch kein Typ IV_B Bogen c zwischen den beiden liegen, so dass es sich um einen zulässigen Pfad aus $\mathcal{FP}$ handelt.

Dienststückdauer

Hier kann man eine analoge Vorgehensweise anwenden. Dazu werden die Typ III Bögen betrachtet, die einen Dienststückwechsel darstellen. Man beachte jedoch, dass ein Typ IV_B Bogen einen neuen Dienstteil und somit ebenfalls ein neues Dienststück einleitet. Ein Teilpfad zwischen einem Typ IV_B Bogen und einem Typ IV_E Bogen, der keinen Typ III Bogen enthält, entspricht demnach auch einem Diensstück.
Durch die Dienststück-Ressourcenbedingung wird, wie im letzten Abschnitt erläutert, die

zulässige Anzahl der Dienststücke modelliert. Für die Restriktionen bzgl. der Dauer dieser Dienststücke seien zu den bereits bekannnten Parametern noch folgende gegeben:

$\lambda_{min}^{Dientstückdauer}$ - minimal zulässige Dauer der einzelnen Dienststücke
$\lambda_{max}^{Dientstückdauer}$ - maximal zulässige Dauer der einzelnen Dienststücke.

Dadurch lassen sich für jeden Bogen $a \in A_{IV_B} \cup A_{III}$, der ein Dienststück einleitet, folgende zulässigen und unzulässigen Bogenmengen definieren :

$$\Pi_a = \{b \in A_{IV_E} \cup A_{III} \,|\, \lambda_{min}^{Dientteildauer} \leq et(b) - st(a) \leq \lambda_{max}^{Dientteildauer}\}$$

$$\overline{\Pi}_a = \{b \in A_{IV_E} \cup A_{III} \,|\, 0 < et(b) - st(a) < \lambda_{min}^{Dientteildauer}\}.$$

Dadurch ergibt sich analog zu den PDCs für Dienstteildauer folgende Klasse von Ungleichungen:

$$\begin{aligned} x_a &\leq \sum_{b \in \Pi_a} x_b \quad \forall\, a \in A_{IV_B} \cup A_{III} \quad (\text{PDC}_\text{I}) \\ x_a + \sum_{b \in \overline{\Pi}_a} x_b &\leq 1 \quad \forall\, a \in A_{IV_B} \cup A_{III} \quad (\text{PDC}_\text{II}). \end{aligned}$$

Der Beweis, dass diese Ungleichungen einerseits zulässig sind und andererseits die Restriktionen bzgl. der Dienststückdauer modellieren, kann mit Hilfe der entsprechenden Mengen Π_a und $\overline{\Pi}_a$ analog zum Beweis für die Dienstteildauer geführt werden.

4.3.2 Verbesserung der Schranke durch PLCs

Dieser Abschnitt befaßt sich mit den Ergänzungselementen und deren Positionslogik. So gibt es zum Beispiel Aufgaben, die nur zu Beginn oder Ende eines Dienstes, d.h im ersten bzw. letzten Dienstteil bzw. -stück, erledigt werden müssen. Andere Aufgaben müssen nur zwischen zwei Dienstteilen bzw. -stücken erledigt werden. Um diese positionsabhängigen Restriktionen an die Dienste bzw. deren Knoten modellieren zu können, benötigt man für alle aus Ergänzungselementen induzierten Knoten folgenden typisierten Input:

Typ	Eigenschaft bzgl. der Anzahl
1	nur 0 ist zulässig
2	nur 1 ist zulässig
3	nur 0 und 1 sind zulässig
4	nur $n \geq 2$ ist zulässig
5	nur 0 und $n \geq 2$ sind zulässig [1]
6	nur 1 und $n \geq 2$ sind zulässig
7	beliebig

Das sind alle für die bearbeiteten Instanzen relevanten Typen, wobei die Eigenschaft bzgl. der Anzahl angibt, welche mögliche Anzahl für die folgenden Parameter zulässig ist.

- Typ für vorhergende Dienstteile $p_v^{Dienstteil} \in \{1, 2 \dots 7\}$ $v \in V_E$
- Typ für vorhergende Dienststücke $p_v^{Dienststück} \in \{1, 2 \dots 7\}$ $v \in V_E$
- Typ für nachfolgende Dienstteile $s_v^{Dienstteil} \in \{1, 2 \dots 7\}$ $v \in V_E$
- Typ für nachfolgende Dienststücke $s_v^{Dienststück} \in \{1, 2 \dots 7\}$ $v \in V_E$

Es werden also für jeden Knoten, der von einem Ergänzungselement induziert wurde, diese vier Parameter verwaltet, die die Positionslogik des Knotens eindeutig bestimmen.

[1] Dieser Fall ist der Vollständigkeit halber erwähnt, obwohl für sämtliche bearbeitete Instanzen derartige Ergänzungselemente nicht auftraten.

Anzahl der vorhergehenden Dienstteile

Ähnlich zu der Vorgehensweise bei den PDCs lassen sich für jeden Knoten $v \in V_E$ auch spezielle Bogenmengen bestimmen.

$$\overline{\Pi}_v = \{e \in A_{IV_B} \text{ mit } st(e) < et(v)\}.$$

Diese Menge enthält alle Bögen, die einen Dienstteil vor dem Knoten v einleiten. Je nach Typ kann man dann lineare Bedingungen formulieren, die die Positionslogik abbbilden. Diese wollen wir fortan mit Position Logic Constraints, kurz PLCs, bezeichnen.

$p_v^{Dienstteil}$	zulässige Anzahl	PLCs
1	0	$\sum_{e \in \delta^+(v)} x_e + x_a \leq 1 \quad \forall\, a \in \overline{\Pi}_v$
2	1	$\sum_{e \in \delta^+(v)} x_e + x_a + x_b \leq 2 \quad \forall\, a \in \overline{\Pi}_v,\, b \in \overline{\Pi}_v \backslash \{a\}$
3	0 und 1	$\sum_{e \in \delta^+(v)} x_e + x_a + x_b \leq 2 \quad \forall\, a \in \overline{\Pi}_v,\, b \in \overline{\Pi}_v \backslash \{a\}$ $\sum_{e \in \delta^+(v)} x_e \leq \sum_{e \in \overline{\Pi}_v} x_e$
4	n	$n \sum_{e \in \delta^+(v)} x_e \leq \sum_{e \in \overline{\Pi}_v} x_e$
5	0 und n	- [2]
6	1 und n	$\sum_{e \in \delta^+(v)} x_e \leq \sum_{e \in \overline{\Pi}_v} x_e$
7	beliebig	-

Anzahl der nachfolgenden Dienstteile

Die Menge Π_v enthält alle Bögen, die einen Dienstteil nach dem Knoten v einleiten.

$$\Pi_v = \{e \in A_{IV_B} \text{ mit } et(v) < st(e)\}.$$

Damit ergeben sich folgende PLCs für die jeweiligen Knotentypen:

[2] Dieser Fall kann nicht mit Hilfe der ISC oder FSC modelliert werden. Es bietet sich jedoch die Möglichkeit solche Ergänzungselemente durch zwei Knoten des Types 1 und 4 zu repräsentieren.

$s_v^{Dienstteil}$	zulässige Anzahl	PLCs
1	0	$\sum_{e \in \delta^+(v)} x_e + x_a \leq 1 \quad \forall\, a \in \Pi_v$
2	1	$\sum_{e \in \delta^+(v)} x_e + x_a + x_b \leq 2 \quad \forall\, a \in \Pi_v,\, b \in \Pi_v \backslash \{a\}$
3	0 und 1	$\sum_{e \in \delta^+(v)} x_e + x_a + x_b \leq 2 \quad \forall\, a \in \Pi_v,\, b \in \Pi_v \backslash \{a\}$ $\sum_{e \in \delta^+(v)} x_e \leq \sum_{e \in \Pi_v} x_e$
4	n	$n \sum_{e \in \delta^+(v)} x_e \leq \sum_{e \in \Pi_v} x_e$
5	0 und n	-
6	1 und n	$\sum_{e \in \delta^+(v)} x_e \leq \sum_{e \in \Pi_v} x_e$
7	beliebig	-

Anzahl der vorhergehenden Dienststücke

Analog zu der Vorgehensweise bei den PLCs für die Positionslogik der Dienstteile lassen sich für jeden Knoten $v \in V_E$ auch spezielle Bogenmengen für die Dienststücke bestimmen:

$$\overline{\Pi}_v = \{e \in A_{IV_B} \cup A_{III} \text{ mit } st(e) < et(v)\}.$$

Diese Menge enthält alle Bögen, die ein Dienststück vor dem Knoten v einleiten.

$p_v^{Diensstück}$	zulässige Anzahl	PLCs
1	0	$\sum_{e \in \delta^+(v)} x_e + x_a \leq 1 \quad \forall\, a \in \overline{\Pi}_v$
2	1	$\sum_{e \in \delta^+(v)} x_e + x_a + x_b \leq 2 \quad \forall\, a \in \overline{\Pi}_v,\, b \in \overline{\Pi}_v \backslash \{a\}$
3	0 und 1	$\sum_{e \in \delta^+(v)} x_e + x_a + x_b \leq 2 \quad \forall\, a \in \overline{\Pi}_v,\, b \in \overline{\Pi}_v \backslash \{a\}$ $\sum_{e \in \delta^+(v)} x_e \leq \sum_{e \in \overline{\Pi}_v} x_e$
4	n	$n \sum_{e \in \delta^+(v)} x_e \leq \sum_{e \in \overline{\Pi}_v} x_e$
5	0 und n	-
6	1 und n	$\sum_{e \in \delta^+(v)} x_e \leq \sum_{e \in \overline{\Pi}_v} x_e$
7	beliebig	-

Anzahl der nachfolgenden Dienststücke

Die Menge Π_v enthält alle Bögen, die ein Dienststück nach dem Knoten v einleiten.

$$\Pi_v = \{e \in A_{IV_B} \cup A_{III} \text{ mit } et(v) < st(e)\}.$$

Damit ergeben sich folgende PLCs:

$s_v^{Dienststück}$	zulässige Anzahl	PLCs
1	0	$\sum_{e\in\delta^+(v)} x_e + x_a \leq 1 \quad \forall\, a \in \Pi_v$
2	1	$\sum_{e\in\delta^+(v)} x_e + x_a + x_b \leq 2 \quad \forall\, a \in \Pi_v,\, b \in \Pi_v \backslash \{a\}$
3	0 und 1	$\sum_{e\in\delta^+(v)} x_e + x_a + x_b \leq 2 \quad \forall\, a \in \Pi_v,\, b \in \Pi_v \backslash \{a\}$ $\sum_{e\in\delta^+(v)} x_e \leq \sum_{e\in\Pi_v} x_e$
4	n	$n \sum_{e\in\delta^+(v)} x_e \leq \sum_{e\in\Pi_v} x_e$
5	0 und n	-
6	1 und n	$\sum_{e\in\delta^+(v)} x_e \leq \sum_{e\in\Pi_v} x_e$
7	beliebig	-

4.3.3 Verbesserung der Schranke durch IPCs

Als letzte Möglichkeit, unzulässige Pfade zu verbieten, wird ein Konzept übernommen, das beim Lösen von TSP-Problemen [3] eingesetzt wird, nämlich das Generieren von Infeasible Path Constraints, kurz IPCs.

$$\chi_P^T x \leq |P| - 1 \;\; \forall P \in \mathcal{P}\backslash\mathcal{D}.$$

Hierbei handelt es sich bei $\chi_P \in \{0,1\}^{|A|}$ um den Bogeninzidenzvektor des Pfades $P \subseteq A$, der wie folgt definiert ist:

$$(\chi_P)_a = \begin{cases} 1, & a \in P \\ 0 & \text{sonst.} \end{cases}$$

Durch die Ganzzahligkeit von x gilt, falls P die Lösung des RCSP ist, $\chi_P^T x = |P|$. Somit kann die Lösung, durch das Hinzufügen der entsprechenden IPC, abgeschnitten und die Schranke im günstigsten Fall verbessert werden. Natürlich beruht hier die Hoffnung darauf, dass nur eine kleine Teilmenge, die wir fortan mit $\mathcal{IP}$ bzeichnen wollen, der Menge $\mathcal{P}\backslash\mathcal{D}$ benötigt wird, um die Schranke entscheidend zu verbessern.
Man benötigt diese Ungleichungen, um die nicht linearen Eigenschaften der Dienste zu modellieren. Auch Pfade, die beispielsweise die Relaxierungen der Pausenregelungen erfüllen, aufgrund der Anrechnungsregel jedoch keinen Dienst darstellen, können somit verboten werden.
Es lassen sich damit aber auch Teilpfadeigenschaften modellieren. Beispielsweise gibt es Restriktionen für die ununterbrochene Lenkzeit eines Dienstes. Verletzt die Pfadlösung der RCSP Relaxierung diese Bedingung, so kann mit Hilfe einer IPC der entsprechende Teilpfad verboten werden, weil er in keinem Dienst dieser Dienstart als zulässiger Teilpfad auftreten kann.
Diese Verfahrensweise adaptiert das Prinzip der k-Shortest Path Algorithmen. Jedoch ist durch die zahlreichen Ressourcenbedingungen und den Komplex der PDCs und PLCs die Pfadmenge erheblich eingeschränkt, so dass diese Strategie, um letztendlich die Lücke zwischen dem Minimum der Relaxierung und dem Minimum der entsprechenden Dienstmenge zu schließen, als sinnvoll betrachtet werden kann.

[3] Es handelt sich hierbei um die bereits in Kapitel 2 vorgestellten Subtour-Elimination Constraints, die Zykel verbieten.

4.4 Algorithmus zum Bestimmen der RCSP Lower Bound

Nachdem nun mögliche Klassen von Ungleichungen zur Verschärfung der Relaxierungen vorgestellt und die Möglichkeit der Pausenregelpartitionierung genau erläutert wurden, folgt in diesem Teil der Arbeit die algorithmische Umsetzung dieser Ideen.

Bestimmen der Unteren Schranke des MLP

Input: Menge der Dienstarten $\mathcal{DT}$
Menge der Pausenregeln $\mathcal{BR}$
Dienstplanungsgraph $D = (V, A)$
Optimalwert des RMLP z
Output: δ und Untere Schranke für das MLP

1: Starte mit $\delta = 0.00$.
2: Für alle Dienstarten und zugehörigen
Pausenregelungenrelaxierungen führe folgendes aus :

3: Erstelle RCSP-MIP$_{dt,br}$ Relaxierung.
4: Bestimme mit Hilfe des Delta-Algorithmus $\delta_{dt,br}$.
5: Falls $\delta_{dt,br} > \delta$
Setze $\delta = \delta_{dt,br}$

6: Gib δ und $\frac{z}{1+\delta}$
als Untere Schranke des MLP aus.

Abbildung 4.5: Verfahren zur Berechnung der RCSP Lower Bound.

Dabei wird im vierten Schritt der folgende Delta-Algorithmus verwendet:

Delta-Algorithmus

Input: Dienstplanungsgraph $D = (V, A)$
Dienstart $dt \in \mathcal{DT}$
Pausenregel $br \in \mathcal{BR}$
RCSP-MIP Instanz
Approximationsgenauigkeit $\epsilon > 0$
Output: minimales δ_{dt}, so dass das
$(\delta - PRICE)$-Problem für dt keine Lösung besitzt

1: Starte RCSP-MIP Algorithmus.

2: Falls Antwort Nein
Gib $\delta_{dt,br} = 0.00$ aus.

4: Andernfalls
5: Initialisiere δ_0.
6: Finde $\delta_{dt,br}$, so dass RCSP-MIP Algorithmus $|c(P)| < \epsilon$ liefert.
7: Gib δ_{dt} aus.

Abbildung 4.6: Algorithmus zur Berechnung des δ.

Der sechste Schritt muss natürlich noch weiter erläutert werden. Als erste Standardlösung kommt eine binäre Suche für δ_{dt} im Intervall $(0, \infty)$ in Frage. Dies kann jedoch zu einer erheblichen Anzahl neuer Aufrufe des RCSP-MIP Algorithmus mit geänderten Kosten führen. Um diese Anzahl zu verringern, stellt sich folgende Idee als praktikabel heraus.
Zu Beginn der Initialisierung, Schritt 5, gilt für den aktuellen optimalen Pfad P mit Kosten $c(P)$, für die Lösung π des aktuellen RDLP und die zugehörigen Lösungen $x^\star$ und $s^\star$:

$$c^T x^\star + d^T s^\star - \sum_{j=1}^{m} \sum_{ij \in A} \pi_i x_{ij}^\star = c(P) < 0$$

Dann sei $\delta_0 > 0$ so definiert, dass $(1 + \delta_0)(c^T x^\star + d^T s^\star) - \sum_{j=1}^{m} \sum_{ij \in A} \pi_i x_{ij}^\star = 0$ gilt.
Dieses δ_0 ist bereits eine untere Schranke für das gesuchte δ_{dt}, da P ein zulässiger Pfad ist. Sollte bereits nach der nächsten Aktualisierung und Optimierung des RCSP $|c(P)| < \epsilon$ [4] gelten, dann ist δ_0 eine ausreichend genaue Lösung unseres Problems. Ansonsten fahren wir analog fort und aktualisieren das δ gemäß des aktuellen neuen Pfades. Diese Technik stellt sich als wesentlich effizienter gegenüber der einfachen binären Suche heraus. Der in Schritt Eins und Sechs verwendete RCSP-MIP Algorithmus funktioniert folgendermaßen:

[4] In den durchgeführten Testrechnungen wurde $\epsilon = 10^{-8}$ verwendet.

RCSP-MIP Algorithmus

Input: Dienstplanungsgraph $D = (V, A)$
Dienstart $dt \in \mathcal{DT}$
Pausenregel $br \in \mathcal{BR}$
Zulässige Cuts C_{dt} für die jeweilige Dienstart dt
maximale Anzahl Branch & Bound Knoten k_{max}

Output: Kosten $c(P) < 0$ oder Antwort Nein.

1: Füge der RCSP-MIP$_{dt,br}$ Instanz C_{dt} hinzu.

2: Löse das MIP mit CPLEX.

3: 1.Fall: Es existiert Lösung P mit Kosten $c(P) < 0$.
Antwort $c(P)$.

4: 2.Fall: Es existiert keine Lösung P mit Kosten $c(P) < 0$.
Antwort Nein.

5: 3.Fall: Branch & Bound Knoten Limit erreicht.**?**
lb = gegenwärtige untere Schranke von CPLEX.
Antwort lb.

Abbildung 4.7: Algorithmus zum Lösen des RCSP-MIP.

Der RCSP-MIP Algorithmus ermittelt somit die Lösung der gegenwärtigen Relaxierung. Die Schärfe dieser Relaxierung hängt jedoch auch von der jeweiligen Menge C_{dt} ab. Allgemein besteht diese Menge aus PDCs, PLCs und IPCs. Diese Menge wird iterativ im Optimierungsprozess aufgebaut und je nach Bedarf zum MIP hinzugefügt.
Das Hinzufügen einer IPC, deren Pfad positive Kosten besitzt, wäre nicht sinnvoll, da dieser keine Lösung des Problems darstellt. Der von CPLEX ausgeführte Branch&Bound&Cut Algorithmus zum Lösen des MIP in Schritt 1 wird daher wie folgt modifiziert. Um ein MIP zu lösen, generiert CPLEX einen Baum aus linearen Relaxierungen. Aus diesem Baum wählt der Algorithmus einen Knoten aus, löst die LP Relaxierung und versucht, falls die Lösung der Relaxierung nicht ganzzahlig ist, Ungleichungen (Cuts) hinzuzufügen.
Findet er somit eine neue, bessere, ganzzahlige Lösung, so wird die gegenwärtige ersetzt und möglicherweise redundante Knoten werden aus dem Baum entfernt. Andernfalls wird eine in der Lösung fraktionale Variable bestimmt und für diese ein Branching durchgeführt, d.h. es werden durch die Modifikation der Schranken dieser Variable zwei neue Knoten im Baum er-

zeugt.
Dieser Prozess wird wiederholt, bis alle Knoten im Baum abgearbeitet sind. Die Struktur dieser Vorgehensweise wird beibehalten, lediglich einzelne Schritte werden modifiziert. Es wird, falls am gegenwärtigen Knoten eine ganzzahlige Lösung ermittelt werden konnte, überprüft , ob diese zu einem Dienst korrespondiert. Ist dies nicht der Fall, so wird die entsprechende Ungleichung, die diese Lösung abschneidet, zur Relaxierung hinzugefügt. CPLEX bietet hier die sinnvolle Möglichkeit, diesen Cut lediglich den Relaxierungen hinzuzufügen, die im Teilbaum des aktuellen Knotens liegen.
Je nach Verletzung wird dann die PDC, PLC oder IPC zu der Menge C_{dt} hinzugefügt. In diesem Fall wird des Weiteren die gegenwärtige ganzzahlige Lösung nicht aktualisiert und ein normales Branching (man beachte es existiert keine fraktionale Variable) durchgeführt.
Sollte es sich jedoch bei der ganzzahligen Lösung um einen Dienst handeln, so wird dieser der Dienstmenge hinzugefügt und die gegenwärtige ganzzahlige Optimallösung aktualisiert. Vorteil dieser Modifikationen ist einerseits, dass als Lösung tatsächlich ein zulässiger Dienst mit negativen und minimalen Kosten ermittelt werden kann, oder andererseits gezeigt werden kann, dass ein solcher nicht existiert.
Hiermit wird abermals deutlich, warum wir das RCSP Modell mit Hilfe von CPLEX als MIP lösen. Alle während der MIP-Optimierung gefundenen ganzzahligen Lösungen, die zu einem zulässigen Dienst korrespondieren, werden ebenfalls der Dienstmenge hinzugefügt.
Ebenso werden neue Cuts der Menge C_{dt} bestimmt, die insgesamt zu einer Verschärfung der Relaxierung führen.
Die Menge C_{dt} ist pausenregelunabhängig gewählt worden, da sich PDCs und PLCs auf Eigenschaften der jeweiligen Dienstart beziehen und Pfade existieren können, die mehrere Pausenregelungsrelaxierungen erfüllen, jedoch trotzdem keine Dienste darstellen. Um aber diese Cuts nicht bei jeder Pausenregelungdrelaxierung neu zu berechnen, wird der während des gesamten Algorithmus aufgebaute Cut-Pool C_{dt} verwendet.

Branch & Bound & Cut Verfahren

Input: Dienstplanungsgraph $D = (V, A)$
Pausenregel $br \in \mathcal{BR}$
Dienstart $dt \in \mathcal{DT}$
RCSP-MIP Instanz
maximale Anzahl Branch & Bound Knoten k_{max}

Output: Zulässige Cuts C_{dt} für die jeweilige Dienstart dt
Pfad P, der zu zulässigen Dienst korrespondiert,
mit Kosten $c(P) < 0$ oder Antwort Nein.

1: Starte mit leerem Baum und $k = 0$.
2: Füge die LP-Relaxierung des RCSP-MIP als Wurzel des Baumes hinzu.
3: Solange noch nicht markierte Knoten existieren
und $k < k_{max}$:

4: Wähle nicht markierten Knoten aus.
5: Löse LP-Relaxierung.

6: 1.Fall: LP-Optimum ist fraktionale Lösung.
Führe Branching aus.

7: 2.Fall: LP-Optimum ist ganzzahlige Lösung
und zulässiger Dienst.
Füge Dienst hinzu und aktualiere ggf.
beste ganzzahlige Lösung.

8: 3.Fall: LP-Optimum ist ganzzahlige Lösung
und kein zulässiger Dienst.
Füge PDC, PLC oder IPC hinzu und
führe Branching aus.

9: Markiere Knoten und setze $k = k + 1$.

10: 1.Fall Alle Knoten im Baum sind abgearbeitet.
11: Gib beste ganzzahlige Lösung zurück oder
falls keine gefunden wurde Antwort Nein.

12: 2.Fall Es existieren nicht abgearbeitete Knoten im Baum.
13: Bestimme von diesen die schlechteste untere Schranke.
14: Gib diese Schranke als Untere Schranke zurück.

Abbildung 4.8: Branch & Bound & Cut Verfahren.

4.5 Lösen des RCSP Modells

Da nun das RCSP Modell, verschiedene Relaxierungstechniken, und der implementierte Lösungsalgorithmus im Detail vorgestellt wurden, sollen an dieser Stelle die in Kapitel 2 vorgestellten Möglichkeiten des Lösens von RCSP-Instanzen auf ihre Anwendbarkeit hin diskutiert werden. Da im Focus dieser Arbeit eher die Zielsetzung stand, die unteren Schranken des DSP, die durch das Lösen der δ-RCSP-Instanzen gewonnen werden, zu analysieren und die Relaxierungen zu verbessern, wird die folgende Diskussion der Lösungsalgorithmen des RCSP nicht mit Daten belegt.

- **MIP-Lösungsalgorithmen**

  Vor allem der Umgang mit dem im Algorithmus generierten ISC/FSC-Komplex stellt kein Problem dar, ebenso wie der Umgang mit weiteren Schnittebenen. [5] Die nicht zu unterschätzenden Nachteile dieser Algorithmen liegen jedoch ebenfalls auf der Hand.
  Zum einen wird die Struktur des Graphen nur teilweise durch geschickte Kalibrierung der Branch & Bound Strategien ausgenutzt.
  Zum anderen ist die Laufzeit bis ein Optimum gefunden wird, kaum abschätzbar. Man kann jedoch nach ausreichender Laufzeit abbrechen und mit einer unteren Schranke, deren Qualität von der LP-Relaxierung des Problems abhängt, weiterarbeiten .

- **Dynamische Programmierung**

  Theoretisch liegt der Vorteil in der Laufzeit, da voll polynomiale Approximationsschemata, wie in Kapitel 2 vorgestellt, existieren und die durch ausreichend genaue Approximation gewonnenen RCSP Lösungen immer noch zu unteren Schranke für das DSP führen würden.
  Die Verwendung von weiteren Ungleichungen, wie z.B. IPCs, PDCs und PLCs, ist jedoch nicht sinnvoll zu integrieren. Es würde keinen Sinn machen, diese als entartete Ressourcenbedingungen a priori zu formulieren, da dann der Komplex der Resource Constraints enorm wachsen würde. Die Menge der zu konstruierenden Labels wäre somit nicht mehr handhabbar.

- **k-th Shorthest Path Ansätze**

  Zwar kann man für den Lösungspfad problemlos entscheiden, ob die Ressourcenbedingungen und weitere Restriktionen erfüllt werden, jedoch wächst der Graph immens mit steigenden k.
  Allein durch die Restriktivität der Ressourcenbedingungen und die enorme Anzahl weitere Bedingungen der Klasse ISC und FSC kann man erwarten, dass sich das gesuchte k in einem Bereich aufhält, der solche Algorithmen ebenfalls untauglich werden lässt.

[5] In diesem Zusammenhang sei erwähnt, dass CPLEX weitere Schnittebenen (GUB, Cover Cuts, Gomory Fractional Cuts) ggf. hinzufügt.

- **2-Phasen Algorithmen**

  Ein Vorteil ist sicherlich, dass die erste Phase Untere Schranken liefert, die man bereits verwenden könnte. Da diese Phase die Laufzeit nicht dominiert, ist dies sicherlich sinnvoll und würde in dieser Betrachtung zu den relativ gesehen schnellsten Unteren Schranken führen. Jedoch kann man ohne exakte Pfadlösung keine weiteren ISCs und FSCs hinzufügen, und somit ist zu erwarten, dass die generierte Schranke im allgemeinen sehr viel schlechter ist als diejenige, die man durch das Lösen mit MIP Lösungsalgorithmen gewinnt.

Letztendlich ist die einzig anwendbare Möglichkeit, die RCSP Relaxierungen zu lösen und vor allem zu verschärfen ein, wie im vierten Kapitel vorgestellter, Branch & Bound & Cut Algorithmus. Alle anderen RCSP-Algorithmen sind nicht geeignet für die aus Dienstplanungsproblemen des ÖPNV entstandenen RCSP-Instanzen.
Begründet ist dies vor allem darin, dass zum einen prinzipiell eine feste Anzahl von relevanten und restriktiven Ressourcenbedingungen existieren (siehe 4.1) und zum anderen , dass einzelne Pfade durch spezielle vorgestellte Verletzungen der Dienstbildungsregeln nicht zulässig sind. Diese Vielzahl von Bedingungen und speziellen Eigenschaften führt zu einer Menge von ISCs und FSCs, die aufgrund ihrer enormen Anzahl nur implizit in die Optimierung einfließen können.
Das Ziel muss es sein Modelle für die einzelnen Dienstarten und Pausenregelungen zu entwickeln, die vollständig auf den Komplex der ISC und FSC verzichten können. Erst dann ist es sinnvoll für die RCSP Instanzen schnellere 2 Phasen Algorithmen einzusetzen.

4.6 Branch & Bound Strategien

Für das Lösen von ganzzahlig gemischten Programmen ist die Wahl der Parameter der Solver, in unserem Fall CPLEX 8.0 [I02], von nicht zu unterschätzender Bedeutung. Deshalb beschäftigt sich dieser Abschnitt kurz mit verschiedenen problemspezifischen Branch&Bound Strategien. Die Auswahl des nächsten Branch&Bound Knoten spielt dabei eine entscheidende Rolle, deshalb wurden folgende Möglichkeiten getestet:

- DFS - Depth First Search (Tiefensuche), d.h. der neueste noch aktive Knoten wird ausgewählt.
- BB - Best Bound (Beste Schranke), d.h. der Knoten mit dem besten Zielfunktionswert der LP-Relaxierung wird ausgewählt.

Auch für das Branching kann man verschieden Strategien diskutieren. Die Wahl der Variable und die Schranke, die verändert wird, sind hierbei zu erwähnen.

- K - Kosten des Bogens:

  Diese Auswahlstrategie wählt den Bogen für das Branching aus, dessen Kosten am geringsten sind. Die heuristische Idee dahinter ist, dass man so letzendlich die negativ

bewerteten Bögen zuerst abarbeitetet und so schneller zu der Entscheidung gelangt, dass kein negativ bewerteter ressourcenbeschränkter Pfad im Digraph existiert.
Existiert jedoch so ein negativ bewerteter ressourcenbeschränkter Pfad, dann ist diese Strategie eher nicht zu empfehlen.

- S - Startzeit des Anfangsknotens:

  Die Priorität wird hier auf die Startzeit des Anfangsknotens des Bogen gelegt, dadurch wird der Digraph in gewissen Sinne zeitlich (von links nach rechts) traversiert, dabei wird auch die Struktureigenschaft der Existenz einer topologischen Knotensortierung im Digraphen ausgenutzt.

- T - Typ des Bogens:

  Durch die Ressourcenbedingungen ist offensichtlich, dass manche Bogentypen wichtiger für die Zulässigkeit eines Pfades sind als andere. Prinzipiell sind hierbei Veränderungen der Schranken für Typ III und Typ IV Bögen gegenüber den anderen Bogentypen vorzuziehen.

Die durchgeführten Testrechnungen haben gezeigt, dass es prinzipiell sinnvoll ist zuerst die untere Schranke einer Bogenvariable zu ändern, d.h. man betrachtet das Subproblem, in dem jeder Pfad den entsprechende Bogen enthält. Als Auswahlstrategien hat die Kombination DFS und T zu den vergleichsweise schnellsten Lösungen geführt und wurde deshalb auch für alle vorgestellten Testrechnungen verwendet.

4.7 Dienstteilexpandierter Dienstplanungsgraph

Der letzte Abschnitt hat die zugrundeliegende Vorgehensweise zur Bestimmung der unteren Schranken dargelegt und auch gezeigt, dass für geteilte Dienste die RCSP Relaxierung schon durch das zugrundeliegende Graphenmodell an Schärfe verliert.
Als Beleg dafür sind zum einen die Dienstteildauerbedingungen zu nennen, die bei nicht geteilten Diensten trivialerweise nicht verletzt sein können, solange die Dienstdauerbedingungen erfüllt sind. Zum anderen sind die Relaxierungen der Pausenregeln zu schwach, da keine Zuordnung zu den Dienstteilen getroffen werden kann. Es ist beispielsweise möglich, dass die Quotientenregel für den gesamten Dienst (Pfad), d.h. für alle Dienstteile kumuliert erfüllt ist, jedoch dennoch ein Dienstteil (Teilpfad) existiert, der den erforderlichen Quotienten unterschreitet.
Aus diesem Grund haben wir das Modell für die geteilten Dienste modifiziert und einen alternativen dienstteilexpandierten Dienstplanungsgraphen, wie in 3.1.3 erläutert, konstruiert.
In diesem Graphen ist es nun möglich, für jeden relevanten Knoten und jede Kante zu entscheiden, zu welchem Dienstteil sie gehört. Damit kann man einerseits die Dienstteildauerbedingungen exakt modellieren und andererseits für alle möglichen Kombinationen der Pausenregelungen exakt eine Relaxierung aufstellen. Ein weiterer Vorteil besteht darin, dass die Positionslogik für die Dienstteile bei der Konstruktion des Graphen bereits überprüft werden kann, d.h. nur die Positionslogik für Dienststücke noch verletzt sein kann.

Das alles wird gewährleistet durch eine im ungünstigsten Fall Verdopplung der Knoten- und Kantenmenge auf der einen Seite und eben durch weitere dienstteilabhängige Resource Constraints auf der anderen. Die daraus resultierenden Veränderungen für die Resource Constraints und für die vorgestellten Relaxierungen der Pausenregelungen sind hier kurz für geteilte Dienste, d.h. für den Fall, das exakt zwei Dienstteile existieren sollen, aufgelistet:

4.7.1 Resource Constraints

- **Anzahl der Dienstteile**

$$\sum_{e \in A_{IV_B} \cap A_1} x_e = 1$$

$$\sum_{e \in A_{IV_B} \cap A_2} x_e = 1$$

- **Anzahl der Dienststücke**

  Maximale, minimale und optimale Anzahl der Dienststücke eines Dienstes können, wie in Abschnitt 4.1 ausführlich erläutert wurde, modelliert werden. Des Weiteren können nun aber maximale, minimale und optimale Anzahl der Dienststücke eines Dienstteiles in das Modell integriert werden. Dazu seien folgende Parameter bekannt:

λ_{opt}	- optimale Anzahl der Dienststücke je Dienstteil
λ_{max}	- maximale Anzahl der Dienststücke je Dienstteil
λ_{min}	- minimale Anzahl der Dienststücke je Dienstteil

  Somit muss man für zwei geteilte Dienste folgende Bedingung hinzufügen:

$$\sum_{e \in (A_{III} \cup A_{IV_B}) \cap A_1} x_e + s_+ - s_- = \lambda_{opt}$$

$$\sum_{e \in (A_{III} \cup A_{IV_B}) \cap A_2} x_e + s_+ - s_- = \lambda_{opt}$$

$$0 \leq s_+ \leq \lambda_{opt} - \lambda_{min}$$

$$0 \leq s_- \leq \lambda_{max} - \lambda_{opt}.$$

- Des Weiteren lässen sich die Bedingungen bzgl. der Dienstteildauer direkt durch folgende Resource Constraints modellieren, wobei $r_e, r \in \mathbb{R}_+^{|A|}$.

λ_{opt}	- optimale Dauer eines Dienstteiles
λ_{max}	- maximale Dauer eines Dienstteiles
λ_{min}	- minimale Dauer eines Dienstteiles

$$r_{(u,v)} = \begin{cases} et(v) - et(u) & , (u,v) \in A_I \cup A_{II} \cup A_{III} \cup A_V \\ et(v) - st(v) & , (u,v) \in A_{IV_B} \\ 0 & , (u,v) \in A_{IV_E} \end{cases}$$

$$\sum_{e \in A_1} r_e x_e + s_+ - s_- = \lambda_{opt}$$

$$\sum_{e \in A_2} r_e x_e + s_+ - s_- = \lambda_{opt}$$

$$0 \leq s_+ \leq \lambda_{opt} - \lambda_{min}$$

$$0 \leq s_- \leq \lambda_{max} - \lambda_{opt}.$$

4.7.2 Relaxierungen der Pausenregel

Hier werden einmal mehr die Vorteile der dienstteilexpandierten Modellierung deutlich: Während im alten Modell bei der Kombination von unterschiedlichen Pausenregelungen die Schwierigkeit darin besteht , überhaupt Relaxierungen zu konstruieren und dies letztendlich, da keine Zuordnung zwischen Dienstteil und Knoten getroffen werden kann, im Allgemeinen nicht zu besseren Schranken führt. So lässt sich nun hingegen jede Kombination der vorgestellten Pausenregelungen konstruieren, d.h. es können für zwei geteilte Dienste mit μ zulässigen Pausenregelungen exakt μ^2-viele Relaxierungen konstruiert werden. Gemäss den bereits eingeführten Parametern ergeben sich folgende Bedingungen:

1. **Keine Pausenanrechnung in beiden Dienstteilen**

$$\sum_{e \in A_1} (w_{min})_e x_e \leq W^{dt}_{max}$$

$$\sum_{e \in A_1} (w_{max})_e x_e \geq W^{dt}_{min}$$

$$\sum_{e \in A_2} (w_{min})_e x_e \leq W^{dt}_{max}$$

$$\sum_{e \in A_2} (w_{max})_e x_e \geq W^{dt}_{min}$$

2. **Keine Pausenanrechnung im ersten und Blockpausenregelung im zweiten Dienstteil**

$$\sum_{e \in A_1} (w_{min})_e x_e \leq W^{dt}_{max}$$

$$\sum_{e \in A_1} (w_{max})_e x_e \geq W^{dt}_{min}$$

$$\sum_{e \in A_2} (b_{max})_e x_e \geq BT$$

$$\sum_{e \in A_2} (\overline{b}_{max})_e x_e \geq B$$

3. **Keine Pausenanrechnung im ersten und Quotientenregelung im zweiten Dienstteil**

$$\sum_{e \in A_1} (w_{min})_e x_e \leq W^{dt}_{max}$$

$$\sum_{e \in A_1} (w_{max})_e x_e \geq W^{dt}_{min}$$

$$\sum_{e \in A_2} (b_{max})_e x_e \geq \frac{1}{q} \sum_{e \in A_2} (w_{min})_e x_e$$

4. **Blockpausenregelung im ersten Dienstteil und Quotientenregelung im zweiten Dienstteil**

$$\sum_{e \in A_1} (b_{max})_e x_e \geq BT$$

$$\sum_{e \in A_1} (\overline{b}_{max})_e x_e \geq B$$

$$\sum_{e \in A_2} (b_{max})_e x_e \geq \frac{1}{q} \sum_{e \in A_2} (w_{min})_e x_e$$

5. **Blockpausenregelung im beiden Dienstteilen**

$$\sum_{e \in A_1} (b_{max})_e x_e \geq BT$$

$$\sum_{e \in A_1} (\overline{b}_{max})_e x_e \geq B$$

$$\sum_{e \in A_2} (b_{max})_e x_e \geq BT$$

$$\sum_{e \in A_2} (\overline{b}_{max})_e x_e \geq B$$

6. **Quotientenregelung im beiden Dienstteilen**

$$\sum_{e \in A_1} (b_{max})_e x_e \geq \frac{1}{q} \sum_{e \in A_1} (w_{min})_e x_e$$

$$\sum_{e \in A_2} (b_{max})_e x_e \geq \frac{1}{q} \sum_{e \in A_2} (w_{min})_e x_e$$

Analoge Bedingungen ergeben sich für die drei fehlenden Kombinationen, die durch Vertauschen der Reihenfolge der Dienstteile enststehen. Würde man den Graphen komplett verdoppelt, d.h. auch Knoten bei denen prinzipiell klar ist, dass sie im ersten Dienstteil erledigt werden, besitzen einen Repräsentanten für den zweiten Dienstteil, so würde die Relaxierung (2) denselben Optimalwert liefern wie die Relaxierung, die ensteht, wenn man keine Pausenanrechnung im zweiten Dienstteil und Blockpausenregelung im ersten fordert. Die Relaxierung verliert damit zwar an Schärfe, es reicht dann jedoch aus nur eine der beiden Relaxierungen

zu lösen.
Diese Entscheidung, solche möglichen Symmetrieeffekte einzubauen und auszunutzen, hängt jedoch von der Art der Instanzen ab. In dem Fall, dass sich wenige Knoten a priori zu einem Dienstteil rechnen lassen, ist diese Vorgehensweise sinnvoll, da somit nur $\sum_{k=1}^{\mu} k$ viele Relaxierungen gelöst werden müssen. Kann man andererseits davon ausgehen, dass für eine Vielzahl von Knoten der Dienstteil fixiert ist, so kann ein erheblicher Unterschied durch die Reihenfolge der Pausenregelung auftreten. Dies wiederum würde es rechtfertigen, μ^2-viele Relaxierungen zu lösen.

4.7.3 Dienstteilreihenfolgeproblematik

Durch die Konstruktion der Dienstteile mit Hilfe der Zeitlinie kann im dienstteilexpandierten Graphen folgende Reihenfolgeproblematik auftreten:
Der erste Typ IV_B Bogen a in der Pfadlösung des RCSP gehört zur Menge A_2, d.h. er leitet den zweiten Dienstteil zuerst ein, während der zweite IV_B Bogen auf dem Pfad zur Menge A_1 gehört.
In diesem Fall kann man mit Hilfe der folgenden Part Order Constraint (POC) des Types FSC, solche Pfade für den Bogen a verbieten, indem man fordert, dass für Pfade, die den Bogen a enthalten, bereits der erste Dienstteil eingeleitet wurde.

$$\Pi_a = \{b \in A_{IV_B} \cap A_1 \,|\, st(b) \leq st(a)\}$$

$$x_a \quad \geq \quad \sum_{e \Pi_a} x_e \quad \text{(POC)}$$

4.8 Blockpausenexpandierter Dienstplanungsgraph

Im gesamten Kapitel wird die Idee, die an die Dienste gestellten Anforderungen einerseits im Graphen und andererseits als Resource Constraints zu modellieren, verfolgt. Das Ziel muss es sein eine Relaxierung zu finden, die möglichst viele Eigenschaften der Dienstmenge bereits im Graphen modelliert. Vor allem der Ungleichungskomplex der „degenerierten" Resource Constraints [6] bietet Verbesserungspotentiale. Die Pausenregelungrelaxierungen vernachlässigen beispielsweise komplett die einzuhaltenden Anrechnungsregeln.
Wie bereits vorgestellt, existieren in unserem Modell drei verschiedene Arten der Anrechnung einer Dienstunterbrechung. Wir wollen uns hierbei exemplarisch auf den Fall, dass nur zwei Möglichkeiten existieren, beschränken. D.h., entweder die Unterbrechung wird als obligatorische Pause angerechnet oder es kommt zu keiner Anrechnung der Unterbechung als Pause. [7]
Die Zulässigkeit einer Anrechnung hängt vor allem von der Arbeitszeit vor bzw. nach dieser Unterbrechung ab, hierzu gibt es meist minimale und maximale Restriktionen. Diese Zeitfenster bzw. die Position der Pausen innerhalb eines Dienstes sind im zugrundeliegenden Dienstplanungsgraphen nicht geeignet modellierbar.
In diesem Abschnitt soll deshalb eine Möglichkeit vorgestellt werden, den Graphen für Blockpausenregelungen geeignet zu expandieren. Die Konstruktion soll vorerst am Beispiel einer 1-Block-Pausenregelung für zusammenhängende Dienste [8] vorgestellt werden.
Für diese Dienstart und diese Pausenregelung existieren für jeden Knoten $v \in V_D \cup V_E$ lediglich drei mögliche Zustände:

- $\mathcal{B}$ Die Pausenzeit des Dienstelementbestandteiles, der den Knoten v induziert, wird als erster Pausenblock genutzt.
- $\mathcal{PRE}$ Die Pausenzeit des Dienstelementbestandteiles, der den Knoten v induziert, wird nicht als Pausenblock genutzt und liegt zeitlich vor dem ersten Pausenblock.
- $\mathcal{POST}$ Die Pausenzeit des Dienstelementbestandteiles, der den Knoten v induziert, wird nicht als Pausenblock genutzt und liegt zeitlich nach dem ersten Pausenblock.

Analog zur bisherigen Notation von Knotentypen kann man die Knotenmenge V somit als Vereinigung $V_{\mathcal{PRE}} \cup V_{\mathcal{B}} \cup V_{\mathcal{POST}} \cup V_H$ darstellen.
Gemäß dieser Zustände ergeben sich für jeden Bogen [9] $a \in A \backslash \{A_{IV} \cup A_V\}$ folgende vier, vom Typ der inzidenten Knoten abhängige, relevante Bogentypen: $\mathcal{PRE} \rightarrow \mathcal{B}$, $\mathcal{PRE} \rightarrow \mathcal{PRE}$, $\mathcal{B} \rightarrow \mathcal{POST}$ und $\mathcal{POST} \rightarrow \mathcal{POST}$.
Mit Hilfe dieser Definition kann aus einem Dienstplanungsgraphen $D = (V, A)$ ein neuer expandierter Dienstplanungsgraph $\overline{D} = (\overline{V}, \overline{A})$ mit $|\overline{V}| \leq 3|V|$ und $|\overline{A}| \leq 4|A|$ konstruiert werden.

[6] Schnittebenen der Klassen ISC und FSC.
[7] Die dritte Möglichkeit der Anrechnung als optionale Pause wird in dieser Relaxierung vernächlässigt, da deren Relevanz lediglich für die ununterbrochen Arbeitszeit von Bedeutung.
[8] Für geteilte Dienste folgt Analoges im dienstteilexpandierten Dienstplanungsgraphen.
[9] Um die möglicherweise zulässigen Pausenzeiten, die auf Typ III Bögen liegen, ins Modell zu integrieren, kann jeder Typ III Bogen zu einem Knoten und 2 Bögen expandiert werden.

Diese Definition eines *1-blockpausenexpandierten Dienstplanungsgraphen* $\overline{D}$ lässt, da nun die Arbeitszeit w_a für alle Bögen $a \in \overline{A}$ eindeutig festgelegt ist, Bedingungen bzgl. der Mindest- und Maximalarbeitszeit vor bzw. nach der ersten Blockpause wie folgt zu:

$$\sum_{a \in A \cap (V_{\mathcal{PRE}} \times V_{\mathcal{B}})} x_a = 1 \tag{4.1}$$

$$\sum_{a \in A \cap (V_{\mathcal{B}} \times V_{\mathcal{POST}})} x_a = 1 \tag{4.2}$$

$$\sum_{a \in A \cap (V_{\mathcal{PRE}} \times V_{\mathcal{PRE}})} w_a x_a \leq W_{max}^{PRE} \tag{4.3}$$

$$\sum_{a \in A \cap (V_{\mathcal{PRE}} \times V_{\mathcal{PRE}})} w_a x_a \geq W_{min}^{PRE} \tag{4.4}$$

$$\sum_{a \in A \cap (V_{\mathcal{POST}} \times V_{\mathcal{POST}})} w_a x_a \leq W_{max}^{POST} \tag{4.5}$$

$$\sum_{a \in A \cap (V_{\mathcal{POST}} \times V_{\mathcal{POST}})} w_a x_a \geq W_{min}^{POST} \tag{4.6}$$

Die Bedingung (4.1) und (4.2) modellieren dabei die Existenz einer zusammenhängenden Blockpause ausreichender Länge im Pfad, wobei eine der beiden Gleichungen offensichtlich redundant ist. Die zu erfüllenden Arbeitszeitfenster vor bzw. nach der Blockpause werden durch die Bedingungen (4.3)-(4.6) sichergestellt.
Eine Verallgemeinerung dieser Modellierung für n-Blockpausenregelungen kann wie folgt geschehen:

- n positionsabhängige Zustände für Knoten, deren Pausenzeit als Blockpause genutzt wird.
- n+1 positionsabhängige Zustände für Knoten, deren Pausenzeit nicht als Blockpause genutzt wird.

In Abbildung 4.9 sind die Bogentypen und Knotentypen, die in diesen allgemeinen Fall relevant sind, dargestellt. Mittels dieser Konstruktion kann man aus einem Dienstplanungsgraphen $D = (V, A)$ ein expandierten Dienstplanungsgraph $\overline{D} = (\overline{V}, \overline{A})$ für die n-Blockpausenregelung mit $|\overline{V}| \leq (2n+1)|V|$ und $|\overline{A}| \leq (3n+1)|A|$ konstruieren.

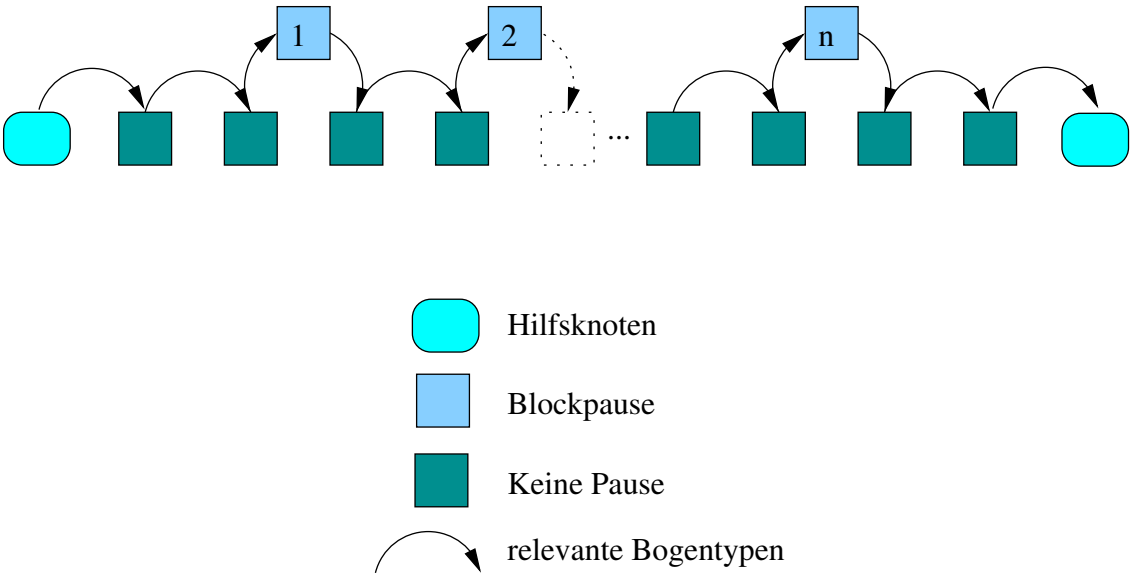

Abbildung 4.9: Relevante Bogentypen für n-Blockpausenregelungexpandierten Dienstplanungsgraphen.

Kapitel 5

Resultate und Rechenergebnisse

In diesem Kapitel sind die erzielten Ergebnisse für die verschiedenen RCSP Relaxierungen praxisnaher Instanzen des ÖPNV zusammengestellt.

5.1 Beschreibung der Instanzen

Der vorgestellte Algorithmus 4.5 zur Berechnung der RCSP Lower Bound wurde anhand von mehreren realen Dienstplanungsinstanzen, die mit Hilfe von DS-OPT 0.97.1 gelöst worden, getestet. Exemplarisch wollen wir die erzielten Ergebnisse für folgende bearbeiteten Instanzen vorstellen. Zu jeder Instanz ist die Anzahl der Dienstelementbestandteile und Ergänzungselemente $|V_D \cup V_E|$, die Anzahl relevanter Knoten [1] $|\overline{V}|$ und die Anzahl der Dienstarten $|\mathcal{DT}|$ in folgender Tabelle dargestellt:

Instanz	$\lvert V_D \cup V_E \rvert$	$\lvert \overline{V} \rvert$	$\lvert \mathcal{DT} \rvert$
IVU 01	423	81	3
IVU 02	1109	185	3
IVU 03	739	185	3
IVU 04	1874	346	3
IVU 05	4507	847	3
IVU 06	6225	1177	3

Abbildung 5.1: Die bearbeiteten Instanzen des ÖPNV.

Bei den Dienstarten handelt es sich um Teilzeitdienste (TZD), zusammenhängende Dienste (ZD) und geteilte Dienste (GD), wobei in den bearbeiteten Szenarien alle vorgestellten Pausenregelungen relevant sind. Für diese wird folgende abkürzende Notation verwendet:

[1] Die Anzahl der Knoten des Graphen, in dem gemäß der Definition der Typ I Bögen Knoten kontrahiert wurden, d.h. diese Anzahl entspricht der Anzahl der Zeilen im SPP ohne Base Constraints.

- 0 keine Anrechnung,
- 1, 2, 3 Blockpausenregelung mit 1, 2 oder 3 Blöcken und
- 6 für Quotientenregelung mit Quotient 6.

5.2 Preprocessing

Problemreduktionsideen, um die Größe des Digraphen zu minimieren, werden in diesem Abschnitt nur informell vorgestellt. Die grundlegenden Ansätze dafür, die Lagrange Multiplikatoren und kürzeste Wege Algorithmen zu nutzen, sind in [BC89] zu finden.
Aufgrund der Definition des Dienstplanungsgraphen kann man keine Reduktion aufgrund der Ressourcen erwarten, da trivialerweise jeder Knoten bzw. Bogen auf mindestens einem ressourcenbeschränkten $(0, n+m+1)$-Pfad liegt, sofern mindestens ein Dienst existiert, der die zugehörigen Dienstelementbestandteile abarbeitet. Die Testläufe, die das belegen, sollen hier deshalb auch nicht explizit vorgestellt werden.
Im Gegensatz dazu ergibt sich aufgrund der Kosten durchaus Reduktionspotential, vor allem bedingt durch die globale obere Schranke von 0. Denn prinzipiell sind wir nur an Pfaden mit negativen Kosten interessiert, so erscheint es sinnvoll, Knoten v aus dem Digraphen zu eleminieren, bei denen die Summe des kürzesten $(0.v)$-Pfades und des kürzesten $(v, n+m+1)$ positiv ist, weil es dann mit Sicherheit kein (ressourcenbeschränkten) $(0, n+m+1)$-Pfad gibt, der diesen Knoten v enthält und negative Kosten besitzt. Diese Betrachtungen lassen sich analog für Bögen und die Lagrange Relaxierung des RCSP durchführen. Man beachte jedoch, dass man diese Preprocessingroutinen lediglich auf sinnvolle Teilmengen der Knoten und Bögen anwenden sollte. [2]
Die Testläufe haben hierbei ergeben, dass es am sinnvollsten ist, lediglich einmal die Knoten $v \in V_E \cup V_D$ zu testen, dazu wurden bereits in DS-OPT berechnete Lagrange Multiplikatoren des RCSP genutzt.
Die Ergebnisse dieser durchgeführten Preprocessingroutine sind in den Abbildungen 5.2-5.7 der Vollständigkeit halber dargestellt, man beachte jedoch, dass aufgrund der veränderten dualen Kosten zwischen den einzelnen Schritten der Column Generation Phase der komplette Dienstplanungsgraph der jeweiligen Dienstart erneut betrachtet werden muss.
Das enorme Reduktionspotential ist anhand des Dienstplanungsgraphen der Instanz IVU 01 für Teilzeitdienste ohne (Abbildung 5.8) und mit Preprocessing (Abbildung 5.9) erkennbar. Obwohl die Visiualisierung für Graphen solcher Größe problematisch ist, verdeutlicht sie nochmals die Notwendigkeit Preprocesingroutinien zu implementieren.
Insgesamt kann man anhand dieser Instanzen zu dem Schluß gelangen, dass sich vor allem für zusammenhängende Dienste dieser Zusatzaufwand lohnt, um die später zu lösenden RCSP Instanzen zu vereinfachen. Die Problematik, dass für geteilte Dienste kaum Knoten entfernt werden können, kann wiederum durch die grössere Anzahl relaxierter Dienstbildungseigenschaften erklärt werden, d.h. vor allem durch die Zeitlinie existieren eine Vielzahl von unzulässigen Pfade, die zu diesen Resultaten führen.

[2] Für Hilfsknoten, Hilfsbögen und Bögen des Types I sind diese Betrachtungen nicht sinnvoll, da deren Redundanz ggf. implizit aus der Redundanz anderer Knoten folgt.

Dienstart	getestete Knoten $v \in V_D \cup V_E$	eliminierte Knoten $v \in V$	benötigte Zeit in sec.
GD	283	61	0,9313
ZD	243	63	0,1863
TZD	243	214	0,1927
GD	283	54	1,2837
ZD	243	59	0,1974
TZD	243	211	0,2448
GD	283	63	0,9085
ZD	243	65	0,1873
TZD	243	227	0,2352
GD	283	63	0,8877
ZD	243	69	0,1863
TZD	243	225	0,1930
GD	283	60	0,8946
ZD	243	66	0,1879
TZD	243	225	0,2397

Abbildung 5.2: Ergebnisse der Preprocessingroutine für Instanz IVU 01.

Dienstart	getestete Knoten $v \in V_D \cup V_E$	eliminierte Knoten $v \in V$	benötigte Zeit in sec.
GD	442	0	0,7352
ZD	529	25	0,6592
TZD	554	86	0,7353
GD	442	0	0,8954
ZD	529	31	0,6752
TZD	554	229	0,7263
GD	442	0	0,7654
ZD	529	21	0,6756
TZD	554	46	0,7365
GD	442	0	0,7922
ZD	529	31	0,6805
TZD	554	381	0,7588
GD	442	0	0,8091
ZD	529	31	0,7335
TZD	554	56	0,8514

Abbildung 5.3: Ergebnisse der Preprocessingroutine für Instanz IVU 02.

Dienstart	getestete Knoten $v \in V_D \cup V_E$	eliminierte Knoten $v \in V$	benötigte Zeit in sec.
GD	442	0	2,3960
ZD	529	32	2,1506
TZD	554	346	2,7670
GD	442	0	2,0677
ZD	529	33	1,9592
TZD	554	463	1,7529
GD	442	0	2,0678
ZD	529	33	1,6631
TZD	554	484	1,8872
GD	442	2	2,1786
ZD	529	33	1,8060
TZD	554	491	2,0132
GD	442	0	3,4792
ZD	529	33	1,6786
TZD	554	421	1,9423

Abbildung 5.4: Ergebnisse der Preprocessingroutine für Instanz IVU 03.

Dienstart	getestete Knoten $v \in V_D \cup V_E$	eliminierte Knoten $v \in V$	benötigte Zeit in sec.
GD	1071	0	3,9996
ZD	1038	117	3,7973
TZD	1020	138	3,0768
GD	1071	0	4,8896
ZD	1038	124	3,2218
TZD	1020	329	3,9804
GD	1071	14	4,5181
ZD	1038	125	3,3238
TZD	1020	238	4,6047
GD	1071	6	6,2162
ZD	1038	126	3,7311
TZD	1020	496	3,5983
GD	1071	0	4,4275
ZD	1038	119	3,3686
TZD	1020	382	3,1424

Abbildung 5.5: Ergebnisse der Preprocessingroutine für Instanz IVU 04.

Dienstart	getestete Knoten $v \in V_D \cup V_E$	eliminierte Knoten $v \in V$	benötigte Zeit in sec.
GD	4166	0	31,2296
ZD	3589	394	21,1865
TZD	3382	385	36,8007
GD	4166	0	19,0886
ZD	3589	350	41,9617
TZD	3382	251	19,0388
GD	4166	0	19,4713
ZD	3589	430	20,8940
TZD	3382	839	14,3729
GD	4166	0	19,3232
ZD	3589	417	30,3582
TZD	3382	1029	30,0112
GD	4166	0	27,3652
ZD	3589	382	21,0355
TZD	3382	571	30,0028

Abbildung 5.6: Ergebnisse der Preprocessingroutine für Instanz IVU 05.

Dienstart	getestete Knoten $v \in V_D \cup V_E$	eliminierte Knoten $v \in V$	benötigte Zeit in sec.
GD	3620	0	24,2170
ZD	3535	228	55,3782
TZD	3471	164	69,2598
GD	3620	0	78,1597
ZD	3535	226	27,5442
TZD	3471	162	59,9392
GD	3620	0	72,7419
ZD	3535	225	38,3252
TZD	3471	159	39,7895
GD	3620	0	60,9219
ZD	3535	233	48,3636
TZD	3471	168	77,1804
GD	3620	0	59,8988
ZD	3535	229	46,9749
TZD	3471	163	62,3483

Abbildung 5.7: Ergebnisse der Preprocessingroutine für Instanz IVU 06.

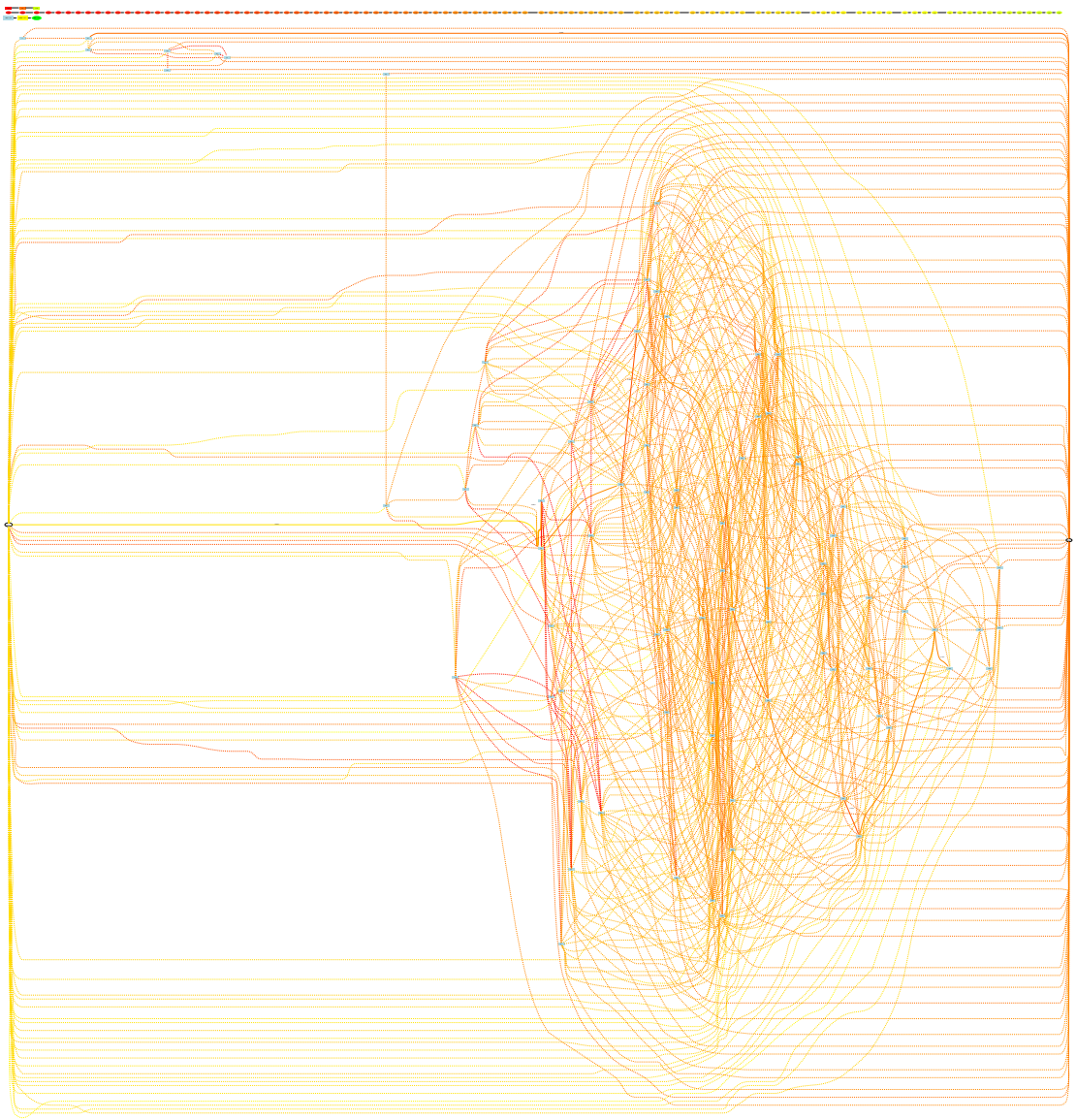

Abbildung 5.8: Dienstplanungsgraph für die Dienstart TZD ohne Preprocessing (IVU 01).

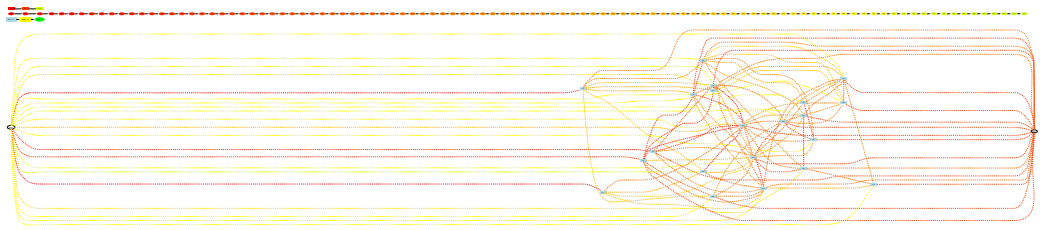

Abbildung 5.9: Dienstplanungsgraph für die Dienstart TZD mit Preprocessing (IVU 01).

5.3 Rechenergebnisse

Für alle Testrechnungen wurde zum Lösen der DSP Instanzen DS-OPT 0.97.1 verwendet und zwischen einzelnen Column Generation Aufrufen die RCSP Lower Bound berechnet. Dazu wurde der vorgestellte Algorithmus 4.5 verwendet. Die folgenden Tabellen und Graphiken sollen vor allem die erzielten Ergebnisse für die unterschiedlichen Relaxierungen vorstellen.
Im Vordergrund steht dabei der Vergleich zwischen der RCSP Relaxierung ohne Berücksichtigung der Pausenregelungen mit den RCSP Relaxierungen für die einzelnen Pausenregelungen. Für alle diese Rechnungen wurde der dienstteilexpandierte Dienstplanungsgraph für die einzelnen Dienstarten separat konstruiert und mit Hilfe der vorgestellten Preprocessingroutine reduziert.

5.3.1 RCSP Lower Bound für verschiedene Relaxierungen

Beschreibung der Testrechnungen

Die Tabellen und Graphiken 5.13-5.21 stellen die erzielten Rechenergebnisse für die Instanz IVU 01 vor. Dazu sind in Abbildung 5.13-5.14 die Ergebnisse der RCSP Relaxierungen der einzelnen Dienstarten im dienstteilexpandierten Graphen ohne Berücksichtigung der Pausenregelungen dargestellt. Die vorgestellten Schnittebenen wurden bei Verletzung der Positionslogik, der ununterbrochenen Lenkzeit, der Diensstückdauer und der Reihenfolge der Dienstteile hinzugefügt. Die in der vierten Spalte erwähnte Anzahl der gefundenen Dienste bezieht sich auf den gesamten Algorithmus 4.5, d.h. jede Lösung des RCSP, die einem zulässigen Dienst entspricht, wird verwertet und dem RMLP hinzugefügt.

Im Vergleich dazu werden in Abbildung 5.15-5.16 die erzielten Resultate der RCSP Relaxierungen der einzelnen Dienstarten im dienstteilexpandierten Graphen mit Berücksichtigung der Pausenregelungen betrachtet. D.h., die vorgestellte Differenzierung nach Pausenregelungen wurde durchgeführt und die dazugehörigen RCSPs ebenfalls mit Hilfe des Algorithmus 4.5 optimal gelöst.

Um die Instanz IVU 01 abschliessend zu analysieren, sind in Tabelle 5.17 die Ergebnisse einer weiteren Testrechnung dargestellt. Dazu wurde ausgehend von der RCSP Relaxierung mit Pausenregelungen jeder nicht mit einem Dienst korrespondierende Pfad durch eine IPC abgeschnitten. Durch diese Vorgehensweise löst man das vom PRICE Problem induzierte Minimierungsproblem exakt, man beachte jedoch, dass die Berechenbarkeit und die Laufzeit mit dieser Technik abhängig von der gewählten Relaxierung ist.
Im Einzelfall kann dies auch für die RCSP Relaxierung ohne Pausenregelung erfolgreich sein. Im Allgemeinen ist diese Relaxierung jedoch zu schwach, d.h die Anzahl der benötigten IPCs ist zu groß, um letztendlich zu einem gültigen Dienst zu gelangen.
Für die Dienstart ZD ist in Abbildung 5.18 exemplarisch der von CPLEX generierte Branch & Bound & Cut Baum visualisiert, wobei keine Pausenregelung betrachtet wurde.
Die Abbildungen 5.19-5.21 zeigen im Gegensatz dazu die Branch & Bound & Cut Bäume unter Berücksichtigung der jeweiligen Pausenregelungen, wobei auf die Relaxierungen der 2 und 3 Blockpausenregelung (siehe Tabelle 5.14) verzichtet wurde.

Diese Visualisierungen sollen vor allem verdeutlichen, dass die Idee mehrere lokale Relaxierungen zu lösen anstatt einer globalen sinnvoll und erfolgreich sein kann.
Jedoch wurden auch Testrechnungen durchgeführt, bei denen die Verschärfung der RCSP Relaxierung durch das Hinuzufügen der relaxierten Resource Constraints für die jeweilige Pausenregelung zu komplexeren Branch&Bound&Bäumen geführt hat.

Analoge Testrechnungen wurden für die Instanz IVU 03 durchgeführt. Die Ergebnisse sind in den Abbildungen 5.22-5.26 dargestellt und stellen repräsentative Werte für die bearbeiteten Instanzen vergleichbarer Größe dar.

Die erzielten Werte der Testläufe für die Instanz IVU 05 sind in Abbildung 5.27-5.29 dargestellt. Aufgrund der Grösse der Instanz wurden nur drei RCSP Lower Bound Berechnungen unter Berücksichtigung der Pausenregelungen durchgeführt.

Laufzeit

In den folgenden Tabellen 5.10 und 5.11 sind die Rechenzeiten der einzelnen Testläufe[3], die auf einem Intel(R) XEON(TM) CPU 2.40GHz mit 3GB Arbeitsspeicher durchgeführt wurden, dokumentiert. Dabei steht in der vierten Spalte die benötigte Gesamtzeit aller Aufrufe des Algorithmus 4.5. Die von CPLEX benötigte Zeit, um die RCSP Instanzen optimal zu lösen, dominiert dabei die benötigte Laufzeit.

Instanz	RCSP Lower Bound Berechnungen	RCSP Berechnungen	benötigte Zeit in sec.	Gesamtzeit in sec.
IVU 01	5	15	19	63
IVU 02	5	15	651	797
IVU 03	5	15	646	1177
IVU 04	5	15	792	2062
IVU 05	5	15	5834	11378
IVU 06	5	15	50336	73101

Abbildung 5.10: Laufzeit für die RCSP Relaxierung ohne Berücksichtigung der Pausenregelung.

[3]CPLEX 8.0, Linux version 2.4, gcc version 3.3.1 (SuSE Linux)

Instanz	RCSP Lower Bound Berechnungen	RCSP Berechnungen	benötigte Zeit in sec.	Gesamtzeit in sec.
IVU 01	5	125	242	276
IVU 02	5	125	10946	11133
IVU 03	5	125	8763	9452
IVU 04	5	125	32495	35463
IVU 05	5	125	100910	109870
IVU 06	5	125	-	-[4]

Abbildung 5.11: Laufzeit für die RCSP Relaxierung mit Berücksichtigung der Pausenregelung.

Resultate

In der Tabelle 5.12 sind alle erzielten Ergebnisse nochmals zusammengefasst. Dabei handelt es sich in der zweiten Spalte, um die unteren Schranken für die triviale RCSP Relaxierung ohne weitere Schnittebenen und ohne Berücksichtigung der Pausenregelung.
Dahingegen steht in der dritten Spalte die beste ermittelte RCSP Lower Bound des MLP der jeweiligen Instanz. Diese in Prozent angegebenen Werte wurden zum einen durch mehrere Column Generation Schritte und zum anderen durch mehrere RCSP Lower Bound Berechnungen inklusive weiterer IPCs erzielt.

Instanz	triviale RCSP Lower Bound in %	beste RCSP Lower Bound in %
IVU 01	26.85	0.00
IVU 02	19.52	0.74
IVU 03	13.45	0.49
IVU 04	22.27	9.91
IVU 05	29.84	8.21
IVU 06	29.02	21.11

Abbildung 5.12: Die erzielten unteren Schranken im Vergleich.

[4]Dieser Testlauf wurde nach 36 Stunden abgebrochen.

Iteration	Anzahl der Dienste im RMLP	Optimalwert des RMLP	Gefundene Dienste	δ in %	RCSP Lower Bound
1	7083	13.94111	0	22.64	11.36759
2	7104	13.94111	0	22.64	11.36759
3	7124	13.94111	0	22.64	11.36759
4	7130	13.94111	0	22.64	11.36759
5	7178	13.94111	0	22.64	11.36759

Abbildung 5.13: RCSP Lower Bound ohne Pausenregelung für IVU 01.

Iteration	Dienstart	Optimalwert des RCSP	Bester Wert eines gefundenen Dienstes	Optimalwert des Root Problem	δ in % %
1	GD	-0.1602	-	-0.2079	8.77
	ZD	-0.277	-	-0.2776	22.64
	TZD	0.0000	-	0.0000	0.00
2	GD	-0.1404	-	-0.2461	7.44
	ZD	-0.277	-	-0.2776	22.64
	TZD	0.0000	-	0.0000	0.00
3	GD	-0.1322	-	-0.2517	7.01
	ZD	-0.277	-	-0.2776	22.64
	TZD	0.0000	-	0.0000	0.00
4	GD	-0.1273	-	-0.1999	6.75
	ZD	-0.277	-	-0.2776	22.64
	TZD	0.0000	-	0.0000	0.00
5	GD	-0.1367	-	-0.2536	7.24
	ZD	-0.277	-	-0.2776	22.64
	TZD	0.0000	-	0.0000	0.00

Abbildung 5.14: Resultate der RCSP Relaxierung ohne Pausenregelung für einzelne Dienstarten.

Iteration	Anzahl der Dienste im RMLP	Optimalwert des RMLP	Gefundene Dienste	δ in %	RCSP Lower Bound
1	7083	13.94111	9	8.77	12.81709
2	7102	13.94111	13	7.24	12.99961
3	7128	13.94111	0	4.82	13.30048
4	7129	13.94111	0	4.68	13.31844
5	7129	13.94111	0	4.68	13.31844

Abbildung 5.15: RCSP Lower Bound mit Pausenregelung für IVU 01.

Iteration	Dienstart und Pausenregelung	Optimalwert des RCSP	Bester Wert eines gefundenen Dienstes	Optimalwert des Root Problem	δ in % %
1	GD 0 0	0.0000	-	-0.1368	0.00
	GD 0 1	-0.0073	-	-0.2017	0.41
	GD 0 2	-0.0073	-	-0.1965	0.41
	GD 0 3	-0.0073	-	-0.1429	0.41
	GD 0 6	0.0000	-	-0.1563	0.00
	GD 1 0	-0.0160	-0.0160	-0.2092	0.90
	GD 1 1	-0.1602	-0.0178	-0.2005	8.77
	GD 1 2	-0.1602	-	-0.1931	8.77
	GD 1 3	-0.0580	-	-0.1140	3.13
	GD 1 6	-0.0160	-0.0160	-0.1703	0.90
	GD 2 0	0.0000	-	-0.1685	0.00
	GD 2 1	0.0000	-	-0.1622	0.00
	GD 2 2	0.0000	-	-0.0955	0.00
	GD 2 3	0.0000	-	0.0000	0.00
	GD 2 6	0.0000	-	-0.1369	0.00
	GD 3 0	0.0000	-	-0.1128	0.00
	GD 3 1	0.0000	-	-0.0692	0.00
	GD 3 2	0.0000	-	0.0000	0.00
	GD 3 3	0.0000	-	0.0000	0.00
	GD 3 6	0.0000	-	-0.1088	0.00
	GD 6 0	0.0000	-	-0.2182	0.00
	GD 6 1	-0.1602	-	-0.2163	8.77
	GD 6 2	-0.1602	-	-0.2034	8.77
	GD 6 3	-0.0580	-	-0.0962	3.13
	GD 6 6	0.0000	-	-0.1703	0.00
	ZD 0	0.0000	-	0.0000	0.00
	ZD 1	-0.0273	-0.0273	-0.0910	1.89
	ZD 2	0.0000	-	-0.0257	0.00
	ZD 3	-0.0273	-0.0273	-0.1044	1.89
	ZD 6	-0.0184	-0.0184	-0.1437	1.21
	TZD 0	0.0000	-	-0.0498	0.00
	TZD 1	0.0000	-	0.0000	0.00
	TZD 2	0.0000	-	0.0000	0.00
	TZD 3	0.0000	-	0.0000	0.00
	TZD 6	0.0000	-	0.0000	0.00

Abbildung 5.16: Resultate der RCSP Relaxierung mit Pausenregelung für einzelne Dienstarten für IVU 01.

Iteration	Anzahl der Dienste im RMLP	Optimalwert des RMLP	Gefundene Dienste	δ in %	RCSP Lower Bound
1	7083	13.94111	11	1.89	13.68294
2	7100	13.94111	21	2.35	13.62039
3	7121	13.94111	1	1.90	13.68173
4	7124	13.94111	0	0.00	13.94111

Abbildung 5.17: Resultate der RCSP Relaxierung mit Pausenregelung inklusive aller IPCs.

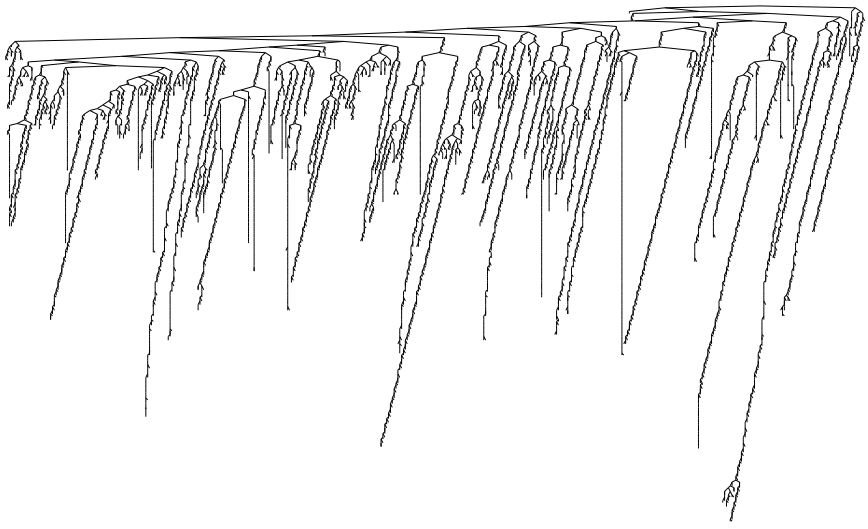

Abbildung 5.18: Branch&Bound&Cut Baum für die RCSP Relaxierung ohne Pausenregelung.

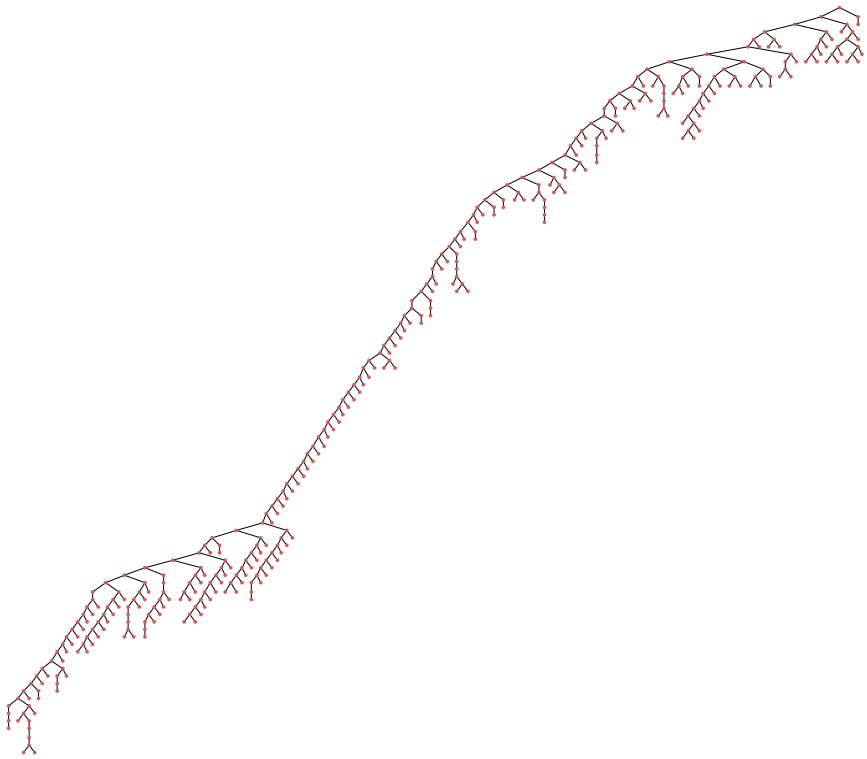

Abbildung 5.19: Branch&Bound&Cut Baum für die RCSP Relaxierung mit Pausenregelung keine Anrechnung.

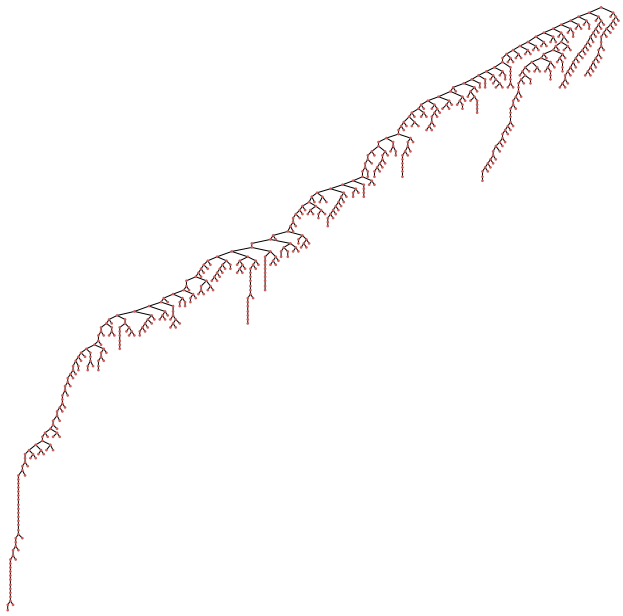

Abbildung 5.20: Branch&Bound&Cut Baum für die RCSP Relaxierung mit Blockpausenregelung.

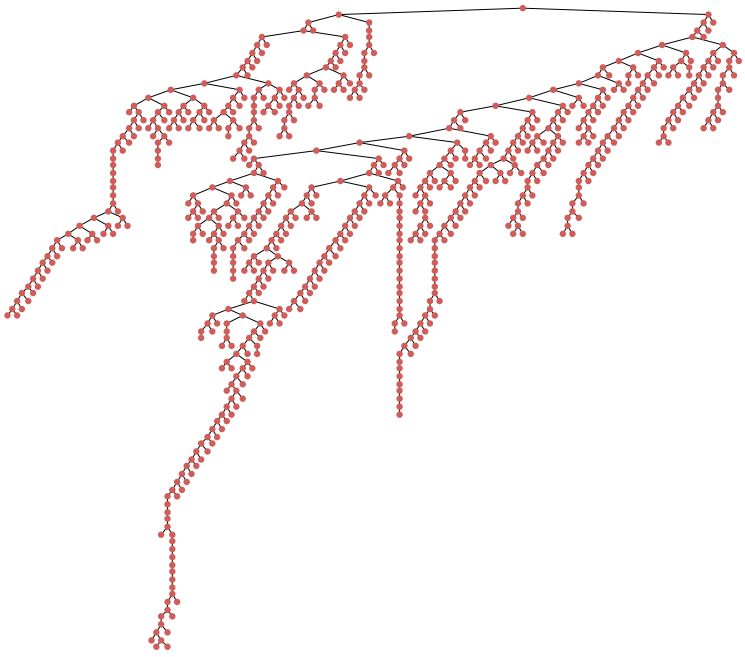

Abbildung 5.21: Branch&Bound&Cut Baum für die RCSP Relaxierung mit Quotientenregelung.

Iteration	Anzahl der Dienste im RMLP	Optimalwert des RMLP	Gefundene Dienste	δ in %	RCSP Lower Bound
1	29886	27.95523	3	13.57	24.61534
2	30400	27.92306	2	7.61	25.94728
3	30491	27.92237	2	8.19	25.80901
4	30575	27.92237	0	6.45	26.22967
5	30620	27.92237	0	7.28	26.02749
6	30651	27.92237	1	6.69	26.17080
7	30746	27.92237	0	7.12	26.06730
8	30776	27.92213	0	6.74	26.15814
9	30793	27.92213	2	6.37	26.25083
10	30816	27.92182	0	6.50	26.21803

Abbildung 5.22: RCSP Lower Bound ohne Pausenregelung für IVU 03.

Iteration	Anzahl der Dienste im RMLP	Optimalwert des RMLP	Gefundene Dienste	δ in %	RCSP Lower Bound
1	29886	27.95523	45	13.57	24.61534
2	30226	27.91782	17	6.57	26.19703
3	30314	27.91719	23	6.44	26.22832
4	30580	27.91719	14	6.45	26.22651
5	30655	27.91719	16	6.28	26.26858
6	30779	27.91719	7	6.56	26.19861
7	30932	27.91719	10	6.56	26.19861
8	31086	27.91719	9	6.39	26.24053
9	31124	27.91504	12	6.82	26.13315
10	31149	27.91456	7	6.14	26.30055

Abbildung 5.23: RCSP Lower Bound mit Pausenregelung für IVU 03.

Iteration	Anzahl der Dienste im RMLP	Optimalwert des RMLP	Gefundene Dienste	δ in %	RCSP Lower Bound
1	32080	27.91402	5	1.88	27.39979
2	32090	27.91399	5	1.77	27.42908
3	32129	27.91042	6	1.80	27.41805
4	32314	27.90463	5	2.00	27.35841

Abbildung 5.24: RCSP Lower Bound mit mehr Column Generation Schritten und Pausenregelung für IVU 03.

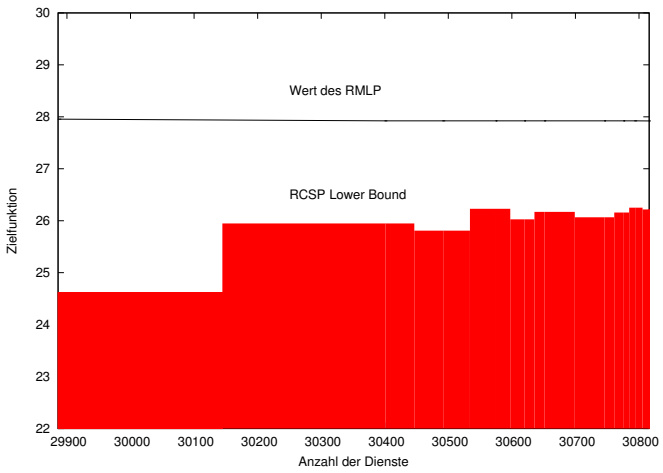

Abbildung 5.25: RCSP Lower Bound ohne Pausenregelung für Instanz IVU 03.

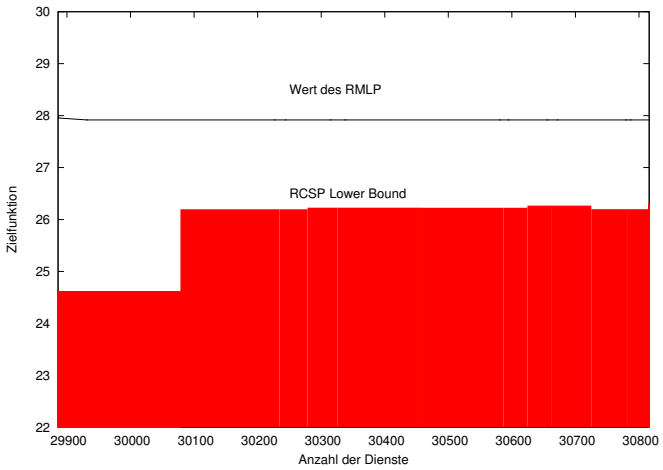

Abbildung 5.26: RCSP Lower Bound mit Pausenregelun für Instanz IVU 03.

Iteration	Anzahl der Dienste im RMLP	Optimalwert des RMLP	Gefundene Dienste	δ in %	RCSP Lower Bound
1	52354	92.45843	16	20.75	76.56910
2	52555	92.42232	11	21.89	75.82561
3	53284	92.41082	11	13.92	81.11831
4	54003	92.37083	4	14.96	80.34858
5	54263	92.35730	10	17.05	78.90503

Abbildung 5.27: RCSP Lower Bound ohne Pausenregelung für IVU 05.

Iteration	Anzahl der Dienste im RMLP	Optimalwert des RMLP	Gefundene Dienste	δ in %	RCSP Lower Bound
1	52354	92.45843	122	20.41	76.78539
2	52555	92.35917	96	21.89	78.48446
3	53284	92.29567	99	13.92	79.33956

Abbildung 5.28: RCSP Lower Bound mit Pausenregelung für IVU 05.

Iteration	Anzahl der Dienste im RMLP	Optimalwert des RMLP	Gefundene Dienste	δ in %	RCSP Lower Bound
1	59692	91.97078	4	10.53	83.21262
2	60157	91.93545	3	13.81	80.78109
3	60769	91.92936	7	9.47	83.97934
4	61193	91.92074	9	8.36	84.82788
5	60769	91.91525	9	8.21	84.94456

Abbildung 5.29: RCSP Lower Bound mit mehr Column Generation Schritten und Pausenregelung für IVU 05.

Auswertung

Die erzielten Resultate für die Instanz IVU 01 spiegeln allgemein das Verhalten für eine Vielzahl kleinerer getester Instanzen wieder. Zum Einen ist die RCSP Lower Bound ohne Pausenregelungen für diese Instanzen erheblich schlechter als die RCSP Lower Bound mit Pausenregelung. Zum Anderen tritt bei der Berechnung der RCSP Lower Bound ohne Pausenregelungen eine Stagnation ein, da meist keine weiteren Dienste generiert werden.
Im Gegensatz dazu werden durch die RCSP Relaxierung mit Pausenregelungen weitere Dienste generiert.
Dennoch sind die RCSP Lower Bounds beider Relaxierungen noch entfernt von der exakten PRICE Lösung. Die Ursache dafür ist, dass die tatsächlichen Regeln zur Pausenbildung lediglich relaxiert im System integriert werden können.
Auch das exakte Lösen des PRICE Problemes führt meist zu einer Vielzahl von Diensten mit reduzierten Kosten, die jedoch zu keiner Verbesserung des RMLP führen.
Dieser Sachverhalt wurde bereits anhand eines Beispiels in Kapitel 3 angesprochen. Die durchgeführten Testläufe belegen letztendlich, dass auch für reale Instanzen des ÖPNV diese Problematik eintritt. Dies ist vor allem bedingt durch eine Vielzahl von Symmetrieeffekten für reale Dienstplanungsinstanzen.
Durch das Lösen des RCSP mit einem Branch&Bound&Cut Algorithmus ist es möglich durch das implizite Hinzufügen aller IPCs, letztendlich den Dienst mit minimalen reduzierten Kosten zu berechnen. Für Instanzen mit $|\overline{V}| \leq 200$ kann dies noch in annehmbarer Rechenzeit geschehen, dazu sei auf Abbildungen 5.10-5.11 verwiesen. Dadurch kann insgesamt gezeigt werden, dass die zugrundeliegende Lösung des RMLP für kleine Instanzen auch optimal für das MLP ist. Das bedeutet insbesondere, dass das Optimierungstool DS-OPT für die Optimierung des MLP keine relevanten Dienste vernachlässigt.

Die Instanz IVU 03 steht für die Klasse der Instanzen mit relevanter Knotenanzahl zwischen 200 und 400. Man beachte, dass die dazu gelösten RCSP Instanzen auch die Ergännzungselemente betrachten müssen, d.h. für den dienstteilexpandierten Dienstplanungsgraphen mit 2 Dienstteilen $D = (V, A)$ gilt: $|V_E| + |\overline{V}| \leq |V| \leq 2(|V_E| + |\overline{V}|)$.
In diesen Szenarien wurden keine deutlichen Verbesserungen durch die Differenzierung nach Pausenregelungen erzielt. Sowohl die RCSP Lower Bound ohne als auch die mit Berücksichtigung der Pausenregelungen erzielte untere Schranke liefern Werte zwischen $3\% \sim 7\%$.
Durch das implizite Hinzufügen aller IPCs werden hier mit der exakten PRICE Lösung Schranken generiert, die eine Differenz zum Optimum von $1\% \sim 2\%$ prognostizieren. Jedoch führen die Dienste, die dadurch generiert werden, nur zu minimalen Verbesserungen des RMLP.
Durch diese unteren Schranken des MLP ist gezeigt, dass die Lösung des RMLP bereits zu Beginn der RCSP Lower Bound Berechnung höchstens $1\% \sim 2\%$ vom Optimalwert des MLP entfernt ist.

Insgesamt kann man mit der RCSP Relaxierung für kleine und mittlere Instanzen sehr gut untere Schranken für das MLP berechnen. Jedoch muss die Relaxierung erst durch geeignete Schnittebenen verschärft werden. Welche Klassen von Ungleichungen letztendlich zu den Verbesserungen der Schranke führen, hängt maßgeblich von der Instanz ab. Das Berechnen der RCSP Lower Bound ohne die vorgstellten ISC und FSC führt jedoch zu keinen sinnvollen

Schranken [5] für die bearbeiteten Instanzen. Aus diesem Grund wurde auf das Vorstellen der Resultate für die RCSP Relaxierung ohne FSC und ISC verzichtet.

Für große Instanzen sind derartige Resultate ohne Weiteres nicht zu erzielen. Einerseits ist der zugrundeliegende dienstteilexpandierte Dienstplanungsgraph, in dem die RCSPs gelöst werden, erheblich größer. Anderseits bieten die aktuellen Lösungen des RMLP im Vergleich noch mehr Verbesserungpotential.[6] Erst, wenn all diese degenerierten Dienste hinzugefügt wurden, kann die RCSP Relaxierung aussagekräftige Schranken liefern.

Das Ziel dieser Arbeit ist es, die Frage zu beantworten, ob es sinnvoll ist RCSP Relaxierungen zu konstruieren, um untere Schranken für DSP Instanzen des ÖPNV zu generieren.
Die erzielten Resultate geben eine positive Antwort auf diese Frage, da die RCSP Lower Bound im Vergleich zu trivialen unteren Schranken erheblich besser ist.
Aufgrund der Laufzeit ist jedoch von einer Integration in die tatsächlichen DSP-Algorithmen abzusehen. Denn dazu müsste der RCSP Algorithmus erheblich beschleunigt werden, was aufgrund der vorgestellten Schnittebenen, die dann explizit als Resource Constraints formuliert werden müssten, kaum möglich ist.
Es bietet sich daher vor allem der Einsatz als Zertifikator nach der Planungsphase an.

5.3.2 Resultate für Blockpausenexpandierten Graphen

Der in Kapitel 4.8 eingeführte blockpausenexpandierte Dienstplanungsgraph wurde für den Fall zusammenhängender Dienste mit $n = 1$ Blöcken implementiert und anhand der vorgestellten Instanzen getestet.
Die dabei erzielten Rechenergebnisse für die Instanz IVU 01 werden in folgenden Abschnitt kurz thematisiert.

Iteration	Dienstart	Optimalwert des RCSP	Bester Wert eines gefundenen Dienstes	Optimalwert des Root Problem	δ in % %
1	ZD	-0.0273	-0.0273[7]	-0.0910	1.89
2	ZD	-0.0767	-0.0767	-0.1087	5.52
3	ZD	0.0000	-	-0.0963	0.00
4	ZD	0.0000	-	-0.0993	0.00
5	ZD	0.0000	-	-0.0993	0.00

Abbildung 5.30: Resultate der RCSP Relaxierung der 1-Block Pausenregelung für zusammenhängende Dienste durch Resource Constraints.

[5] Der Wert für δ betrug teilweise zwischen 50 % und 200 %. Im Vergleich dazu wurden mit der LP-Relaxierung des RCSP mit allen beschriebenen FSC und ISC bessere Ergebnisse erzielt.

[6] Daraus resultieren auch die im Vergleich schlechteren Preprocessingergebnisse.

[7] Man beachte, dass ein Dienst, der diese Relaxierung erfüllt, nicht unbedingt die 1-Block Pausenregelung als Parameter besitzen muss. In diesem Beispiel erfüllen die gefundenen Dienste lediglich die Quotientenregelung.

Iteration	Dienstart	Optimalwert des RCSP	Bester Wert eines gefundenen Dienstes	Optimalwert des Root Problem	δ in % %
1	ZD	0.0000	-	-0.0416	0.00
2	ZD	0.0000	-	-0.0330	0.00
3	ZD	0.0000	-	-0.0240	0.00
4	ZD	0.0000	-	-0.0142	0.00
5	ZD	0.0000	-	-0.0142	0.00

Abbildung 5.31: Resultate der RCSP Relaxierung der 1-Block Pausenregelung für zusammenhängende Dienste im Blockpausenexpandierten Dienstplanungsgraphen.

Insgesamt werden durch die Relaxierungen der 1-Block Pausenregelung im expandierten Dienstplanungsgraphen für alle Instanzen bessere Resultate erzielt. Da jedoch dieses Vorgehen für die Quotientenregel problematisch ist, konnten damit keine Verbesserung der globalen unteren Schranke erzielt werden.

5.3.3 Beobachtungen

Letztendlich kann man nach den gesamten Betrachtungen der unterschiedlichen Relaxierungen der Pausenregelungen zu dem Schluß gelangen, dass vor allem die Quotientenregelung problematisch ist. [8]
Eine einfache Erklärung dafür liefern die Anrechnungsregeln. Für die Pausenregelung „keine Anrechnung“ hat das Relaxieren der Anrechnungsregeln trivialerweise keinen Einfluß, darin begündet sich auch die Stärke dieser Relaxierung. Für die meisten Instanzen kann dadurch relativ schnell gezeigt werden, dass keine Dienste mit dieser Pausenregelung und negativen reduzierten Kosten existieren.
Die Integration der Anrechnungsregel für Blockpausen kann, wie gezeigt, in expandierten Digraphen erfolgen. Für den Fall nur eines Blockes wurde auch dies an mehreren Instanzen getestet und führte zu positiven Ergebnissen.
Dahingegen ist durch das Vernächlässigen der Anrechnungsregel für die Quotientenregel die RCSP Relaxierung dieser Pausenregelung meist zu schlecht.

[8] Ein Beleg dafür ist außerdem, dass die meisten, während der RCSP Berechnung gefundenen, Dienste die Quotientenregelung erfüllen.

5.4 Ausblick

Die Möglichkeit, durch das exakte Lösen von RCSP Instanzen untere Schranken für das DSP zu generieren, stellt sich als durchaus sinnvoll dar. Die erzielten unteren Schranken für das MLP belegen dies.
Je mehr Dienstbildungseigenschaften jedoch durch die Schnittebenen der Klasse FSC und ISC sichergestellt werden müssen, desto komplexer und vor allem langwieriger wird die Berechnung des Optimalwertes des RCSP. Deswegen ist es sinnvoller, wie vorgestellt, expandierte Graphen, in denen mehr Dienstbildungseigenschaften direkt als Resource Constraints modellierbar sind, zu konstruieren.
Somit wäre es möglich schnellere RCSP Lösungsalgorithmen anzuwenden. Jedoch ist dabei zu beachten, dass dann der implizite Einsatz von Schnittebenen problematisch wird und deshalb keinesfalls feststeht, ob so vergleichbare untere Schranken generiert werden können.
Durch die vorgestellten und angewandten Techniken erhält man zudem instanzabhängige Informationen über die Rolle der Dienstarten und Pausenregelungen, d.h. insbesondere welche Dienste womöglich noch zu einem besseren Dienstplan führen könnten. Diese Informationen können durchaus hilfreich für die Column Generation Kalibrierung sein.
Insgesamt wird jedoch dieses Vorgehen, d.h. insbesondere der gesamte Optimierungsprozess der Dienstplanung im ÖPNV, durch die Vielzahl an existierenden Pausenregelungen und Anrechnungsregeln erheblich erschwert.
Zwar ist es durchaus möglich alle relevanten Regelungen in das System zu integrieren, dennoch ist deutlich geworden, dass durch die Existenz eines komplexen und kaum transparenten Regelwerkes im ÖPNV das optimale Lösen von DSP Instanzen äußerst schwierig ist.

Kapitel 6

Anhang

6.1 Mathematische Grundlagen

6.1.1 Graphentheorie

- Ein Graph G ist ein Tripel (V, E, Ψ) bestehend aus einer (endlichen) nicht-leeren Menge V, einer Menge E und einer Inzidenzfunktion $\Psi : E \to V^{(2)}$. Hierbei bezeichnet $V^{(2)}$ die Menge der ungeordneten Paare von (nicht notwendigerweise verschiedenen) Elementen von V. Ein Element aus V heißt Knoten (englisch: vertex), ein Element aus E heißt Kante (englisch: edge). Zu jeder Kante $e \in E$ gibt es also Knoten $u, v \in V$ mit $\Psi(e) = uv = vu$.

- Ein Digraph (oder gerichteter Graph) $D = (V, A)$ besteht aus einer (endlichen) nicht-leeren Knotenmenge V und einer (endlichen) Menge A von Bögen (englisch: arc). Ein Bogen a ist ein geordnetes Paar von Knoten, also $a = (u, v)$, u ist der Anfangs- oder Startknoten, v der End- oder Zielknoten von a, dann heißt u Vorgänger von v, v Nachfolger von u, a inzidiert mit u und v.
  (Der Vollständigkeit halber müßten wir hier ebenfalls eine Inzidenzfunktion $\Psi = (t, h) : A \to V \times V$ einführen. Für einen Bogen $a \in A$ ist dann $t(a)$ der Anfangsknoten (englisch: tail) und $h(a)$ der Endknoten (englisch: head) von a.) Für $W \subseteq V$ sei $\delta^+(W) := \{(i, j) \in A |\ i \in W, j \notin W\}, \delta^-(W) := \delta^+(V \setminus W)$ und $\delta(W) := \delta^+(W) \cup \delta^-(W)$. In dieser Arbeit werden jedoch nur die Mengen $\delta^+(\{v\}), \delta^-(\{v\}), \delta(\{v\})$ benötigt, welche wir fortan mit $\delta^+(v), \delta^-(v), \delta(v)$ abkürzen wollen. Der Außengrad (Innengrad) von v ist die Anzahl der Bögen mit Anfangsknoten (Endknoten) v. Die Summe von Außengrad und Innengrad ist der Grad von v.

- In einem Digraphen heißt eine endliche Folge $P = (v_0, a_1, v_1, a_2, v_2, \cdots, a_k, v_k)$, $k \geq 0$ Kette, falls P mit einem Knoten beginnt, mit einem nicht notwendigerweise gleichen Knoten endet und in der Knoten und Bögen alternierend auftreten, so daß jeder Bogen a_i mit den beiden Knoten v_{i-1} und v_i inzidiert. Falls in einem Digraphen alle Bögen a_i der Kette P der Form (v_{i-1}, v_i) sind, d.h. sie sind gleichgerichtet, so nennt man P gerichtete Kette oder (v_0, v_k)-Pfad. Man beachte hierbei, dass Bögen und Knoten wiederholt auftreten können. Will man dies nicht zulassen, so spricht man von einfachen (simple) bzw. elementaren Pfaden.

- Um die Struktur eines Digraphen vereinfacht in mathematischen Formeln abzubilden, sei folgende Matrix $N \in \{-1, 0, 1\}^{|A| \times |V|}$ als Knoten-Bogen-Inzidenzmatrix des Digraphen $D = (V, A)$ bezeichnet. Dabei gilt:

$$N_{(v,a)} = \begin{cases} 1 & \text{, falls Knoten v Startknoten des Bogens a ist} \\ -1 & \text{, falls Knoten v Zielknoten des Bogens a ist} \\ 0 & \text{, sonst.} \end{cases}$$

- Eine Abbildung $\omega : V \to \{1, 2, \ldots, n\}$ heißt topologische Knotensortierung des Digraphen $D = (V, A)$ mit $|V| = n$, falls für alle $(u, v) \in A$ gilt $\omega(u) < \omega(v)$. Man beachte, dass diese Sortierung keinesfalls eindeutig ist, jedoch gilt die Äquivalenz der folgenden Aussagen:

  1. Der Digraph $D = (V, A)$ besitzt eine topologische Knotensortierung.
  2. Der Digraph $D = (V, A)$ ist zykelfrei.

6.1.2 Optimierung

- Sei folgendes allgemein als Optimierungsproblem definiert:
  Gegeben seien eine Menge S, die wir fortan mit zulässiger Menge bezeichnen wollen, und eine geordnete Menge $(T, \leq)$, dann gilt zwischen je zwei Elementen $s, t \in T$ genau eine der folgenden Beziehungen $s < t$, $s > t$ oder $s = t$. Des weiteren sei eine Abbildung $f : S \to T$ gegeben. Gesucht ist ein Element $x^* \in S$ mit der Eigenschaft $f(x^*) \geq f(x)$ für alle $x \in S$ (Maximierungsproblem) oder $f(x^*) \leq f(x)$ für alle $x \in S$ (Minimierungsproblem). Hierfür werden folgende Schreibweisen benutzt:

$$\begin{array}{lll} \max\limits_{x \in S} f(x) & \text{oder} & \max\{f(x) \mid x \in S\}, \\ \min\limits_{x \in S} f(x) & \text{oder} & \min\{f(x) \mid x \in S\}. \end{array}$$

- Gegeben seien $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{(m,n)}$, $b \in \mathbb{R}^m$, dann heißt

$$\begin{array}{lll} \text{(LP)} & \max & c^T x \\ & s.t. & Ax \leq b \\ & & x \in \mathbb{R}^n \end{array}$$

  lineares Optimierungsproblem. Es stellt hierbei keinerlei Einschränkung dar, dass es sich um Ungleichungen handelt, da offensichtliche Transformationen zu Gleichungen existieren. Mit (DP) wollen wir das zu diesem LP duale Lineare Programm bezeichnen, welches sich aufgrund der Sätze der Dualitätstheorie bei der Optimierung und der Beweisführung als hilfreich erweist.

$$\begin{array}{lll} \text{(DP)} & \min & b^T y \\ & s.t. & A^T y = c \\ & & y \in \mathbb{R}^n_+ \end{array}$$

- Gegeben seien $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{(m,n)}$, $b \in \mathbb{R}^m$, dann heißt

$$\begin{array}{lll} \text{(IP)} & \max & c^T x \\ & s.t. & Ax \leq b \\ & & x \in \mathbb{Z}^n \end{array}$$

lineares ganzzahliges (oder kurz: ganzzahliges) Optimierungsproblem.

- Sind sowohl ganzzahlige, als auch reelle Variablen im Problem vorhanden, so spricht man von einem gemischt ganzzahligen linearen Programm (MIP).
- Eine Relaxierung zu einem gegebenen Minimierungsproblem $\min\{\overline{f}(x) \mid x \in \overline{S}\}$ ist ein weiteres Minimierungsproblem $\min\{f(x) \mid x \in S\}$ für das die folgenden zwei Aussagen gelten:
  1. $\overline{S} \subseteq S$,
  2. $\overline{f}(x) \geq f(x)$.
- Ein Entscheidungsproblem Π, d.h. ein Problem, dass nur die Antworten 'ja' oder 'nein' zulässt, gehört zur Klasse $\mathcal{NP}$, wenn es die folgenden Eigenschaften aufweist.
  1. Für jede Instanz $I \in \Pi$, für die die Antwort 'ja' lautet, gibt es mindestens ein Objekt Q, mit dessen Hilfe die Korrektheit der 'ja'-Antwort überprüft werden kann.
  2. Es gibt einen Algorithmus, der die Instanz $I \in \Pi$ und ein Zusatzobjekt Q als Input akzeptiert und der in einer Laufzeit, die polynomial in der Kodierungslänge von I ist, überprüft, ob Q ein Objekt ist, aufgrund dessen Existenz eine 'ja'-Antwort für T gegeben werden muss.

  Ein Entscheidungsproblem Π heißt $\mathcal{NP}$-vollständig, falls $\Pi \in \mathcal{NP}$ und falls jedes andere Problem aus $\mathcal{NP}$ polynomial in Π transformiert werden kann. Hingegen heißt ein Optimierungsproblem Π $\mathcal{NP}$-schwer, falls das dazugehörige Entscheidungsproblem, welches die Frage nach der Existenz einer Lösung grösser bzw. kleiner als eine gegebene Schranke ist, $\mathcal{NP}$-vollständig ist.
- Für viele relevante Probleme existieren jedoch keine polynomialen und somit effizienten Lösungsalgorithmen. Um aber solche Probleme trotzdem bearbeiten zu können, begnügt man sich mit approximativen Lösungen. Ein Algorithmus, der in polynomialer Zeit eine zulässige Lösung mit der Eigenschaft generiert, das deren Zielfunktionswert höchstens um den Faktor $(1 + \epsilon)$ vom Optimalwert des Problems abweicht, nennt man voll polynomiales ϵ-Approximationsschema, kurz FPAS.

6.1.3 Polyedertheorie

- Ein Vektor $x \in \mathbb{R}^n$ heißt Linearkombination der Vektoren $x_1, \ldots, x_k \in \mathbb{R}^n$, falls es einen Vektor $\lambda = (\lambda_1, \ldots, \lambda_k)^T \in \mathbb{R}^k$ gibt mit

$$x = \sum_{i=1}^{k} \lambda_i x_i \ .$$

  Gilt zusätzlich $\lambda \geq 0$ und $\lambda^T \mathbb{1} = 1$ so heißt x Konvexkombination der Vektoren $x_1, \ldots, x_k$. Diese Kombinationen heißen echt, falls weder $\lambda = 0$ noch $\lambda = e_j$ für ein $j \in \{1, \ldots, k\}$ gilt. Analog bezeichnet $\text{conv}(S)$ für eine Teilmenge $S \subseteq \mathbb{R}^n, S \neq \emptyset$ die konvexe Hülle von S, d.h. die Menge aller Vektoren, die als konvexe Kombination von endlich vielen Vektoren aus S dargestellt werden können.

- Die zulässige Menge eines LPs wird durch den Ungleichungskomplex $Ax \leq b$ definiert. Jede einzelne Ungleichung definiert hierbei einen geschloßenen Halbraum im $\mathbb{R}^n$, während die Lösungen von Gleichungen der Form $a^T x = b$ mit $a \in \mathbb{R}^n \backslash \{0\}$, $b \in \mathbb{R}$ und $x \in \mathbb{R}^n$ einer Hyperebene entsprechen (Abbildung 6.1).

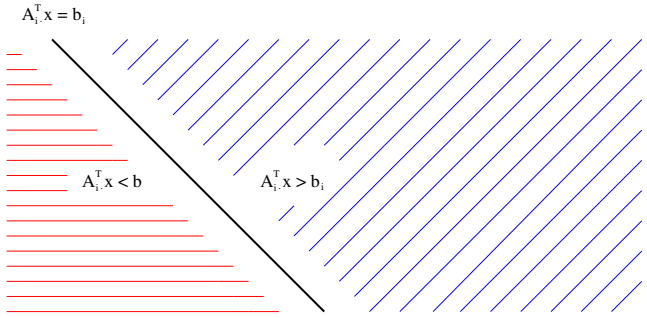

Abbildung 6.1: Hyperebenen und Halbräume.

Die zulässige Menge eines LPs entspricht somit dem Schnitt dieser m geschloßenen Halbräume. Diese Menge $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ bezeichnet man als Polyeder. Ist diese Menge beschränkt, so spricht man von einem Polytop. Andererseits kann man ein Polytop als konvexe Hülle seiner Ecken definieren, d.h. $P = \text{conv}\{v \in V\}$. Die exakte Definition von Ecken wollen wir jedoch nicht einführen und bauen auf die durch die in Abbildung 6.2 gegebene Intuition.

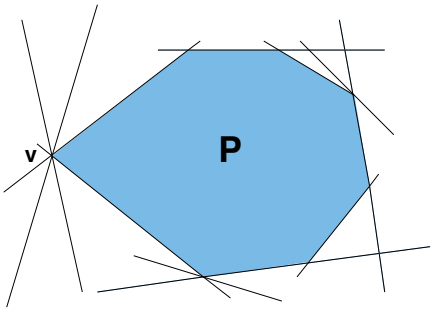

Abbildung 6.2: Polytop mit endlich vielen Ecken.

- Das schnelle Lösen von solchen Linearen Programmen ist eine der grössten und bedeutensten Errungenschaften der Angewandten Mathematik. Algorithmen, die dies leisten, sind Innere Punkte Methoden und Varianten des Simplexalgorithmus. Ausgenutzt wird dabei die Tatsache, dass, falls P nicht leer und nicht unbeschränkt ist, der Optimalwert einer linearen Zielfunktion mindestens in einer Ecke angenommen wird. Deswegen wird die simple Idee verfolgt, von einer Ecke zu einer anderen Ecke zu gelangen, so dass sich der Zielfunktionswert verbessert. Notwendige Bedingung dafür ist jedoch bei Minimierungsproblemen, dass negative reduzierte Kosten existieren. Sind aber alle diese, hier nicht weiter definierten, reduzierten Kosten positiv, so ist die aktuelle Ecke eine Optimallösung. Diese informelle Sichtweise der Lösungsmethodik soll an dieser Stelle ausreichen.

- Um jedoch MIPs optimal zu lösen, werden Branch & Bound & Cut Algorithmen verwendet. Im Allgemeinen ist Branch & Bound eine enumerative Suchstrategie. Das zu bearbeitende Problem sollte dabei die folgenden Eigenschaften besitzen.
  Zum einen sollten untere und obere Schranken relativ 'schnell und leicht' zu berechnen sein (*Bound*), und zum anderen muss man die Menge der zulässigen Lösungen strukturell aufspalten können, um 'kleinere und leichtere' Subproblemen zu konstruieren (*Branch*). Diese Eigenschaften erfüllen gemischt ganzzahlige lineare Optimierungsprobleme, wobei wir uns ohne Beschränkung der Allgemeinheit bei der Erläuterung der Vorgehensweise auf Minimierungsprobleme beziehen. Einerseits liefert die LP Relaxierung des MIP eine untere Schranke und andererseits kann man die zulässige Lösungsmenge geeignet aufspalten, indem man die Beschränkungen einer ganzzahligen Variable verändert. Am Beispiel für $0/1-$Variablen wird dies am besten deutlich, denn in diesem Fall ist die Variable entweder 1 oder 0.
  Die Branch & Bound Algorithmen zum Lösen eines MIPs funktionieren daher wie folgt: Begonnen wird mit dem Ausgangsproblem, welches oftmals mit *root problem* bezeichnet wird. Für dieses werden dann die Schranken generiert, d.h. die LP Relaxierung wird optimal gelöst. In dem Fall, dass keine zulässige Lösung heuristisch ermittelt werden konnte und somit noch keine obere Schranke existiert, wird diese mit $+\infty$ initialisiert und darauf gebaut, dass im Laufe des Algorithmus zulässige Lösungen generiert werden. Man beachte jedoch, dass es durchaus möglich ist, dass überhaupt keine zulässige Lösung existiert. Wenn die untere und obere Schranke übereinstimmen, so ist die zur oberen

Schranke korrespondierende Lösung optimal. Ist dies nicht der Fall, so wird die zulässige Menge sinnvoll aufgespaltet und ein Baum von Subproblemen konstruiert.
Für die so entstandenen Subprobleme, ist das root problem der Vaterknoten im Baum. Der Algorithmus wird nun rekursiv auf einen noch nicht bearbeiteten Knoten im Baum angewandt. Sollte man ein Subproblem im Baum optimal gelöst haben, so hat man damit eine zulässige Lösung für das root problem gefunden, jedoch ist diese nicht notwendigerweise global optimal. Lediglich andere Subprobleme, deren untere Schranke den Wert dieser neuen zulässigen Lösung überschreitet, können aus dem Baum entfernt werden.
Dieser Vorgang wird solange durchgeführt, bis alle Knoten im Branch & Bound-Baum abgearbeitet sind oder bis die Differenz zwischen bester gegenwärtiger Lösung und den unteren Schranken aller unbearbeiteten Subprobleme klein genug ist. Der Nachteil solcher Algorithmen ist die offensichtliche Gefahr, alle Lösungen aufzuzählen (zu enumerieren), was theoretisch exponentieller Aufwand und praktisch endlose Berechnungen bedeuten.
Die Qualität der Schranken ist daher ausschlaggebend für die Größe des zu lösenden Baumes. Deshalb wird versucht, den Bound Schritt wie folgt zu verbessern:
Sollte die Lösung der LP-Relaxierung Ganzzahligkeitsbedingungen verletzen, so erscheint es sinnvoll, Ungleichungen hinzuzufügen, die diese Lösung abschneiden und gleichzeitig für alle ganzzahligen Optimallösungen gültig sind. Diese Ungleichungen werden Schnittebenen genannt und führen dazu, dass man schneller zulässige Lösungen ermittelt und bessere Schranken für die Subprobleme findet (*Cut*).

Zusammenfassung

Die vorliegende Diplomarbeit beschäftigt sich mit einem fundamentalen Problem der Graphentheorie, dem **R**esource **C**onstrained **S**hortest **P**ath Problem (RCSP). Dieses Problem unterscheidet sich nur durch weitere lineare Pfadbedingungen vom einfachen kürzesten Wege Problem in Digraphen. Komplexität, exakte Lösungsalgorithmen und Approximationsschemata werden diskutiert und auf ihre Anwendbarkeit auf ein aus der Praxis stammendes Problem hin untersucht. Die spezielle Anwendung, die dabei auf dieses Problem hinführt, ist die Dienstplanung im ÖPNV und die Frage nach der Qualität der erzielten Ergebnisse.

Das klassische Dienstplanungsproblem besteht darin, einen kostenminimalen Dienstplan aus einer Menge zulässiger Dienste zu generieren. Dies wird graphentheoretisch modelliert und durch das Lösen von **Set Partitioning Problemen** in der Praxis realisiert. Da die genutzten Verfahren nur einen Teil der Dienste berücksichtigen, nämlich jenen, den der sog. Column Generator heuristisch ermittelt hat, fällt es schwer, exakte Aussagen über die Differenz zwischen der berechneten Lösung und dem Optimum zu treffen.
Dies wird in dieser Arbeit thematisiert, indem untere Schranken für das Dienstplanungsproblem durch das exakte Lösen von RCSP-Instanzen produziert werden.

Die Diplomarbeit besteht aus sechs Kapiteln und ist in folgende thematische Abschnitte geteilt.

Der erste Teil (Kapitel 1) führt allgemein in das Thema ein und erläutert die Problemstellung der Dienstplanung im ÖPNV.

Im zweiten Teil (Kapitel 2-3) wird ein Überblick über das RCSP und verschiedenste Lösungsansätze aus der Literatur gegeben. Danach folgt eine ausführliche Erläuterung der mathematischen Modellierung des Dienstplanungsproblems, und erst dann wird die Verbindung zum RCSP aufgezeigt.

Im dritten und letzten Teil (Kapitel 4-5) stehen Möglichkeiten zur Verbesserung der unteren Schranken, die algorithmischen Vorgehensweisen und die dabei erzielten Ergebnisse im Vordergrund.

Im letzten Kapitel werden ausschließlich mathematische Grundlagen, die in dieser Arbeit Verwendung finden, kurz und knapp beschrieben.

Abbildungsverzeichnis

Notation

Symbol	Bedeutung
V	Knotenmenge mit $\|V\| = n$
A	Bogenmenge mit $\|A\| = m$
$D = (V, A)$	Digraph
N	Knoten-Bogen-Inzidenzmatrix
R	Ressourcenmatrix (Resource Constraints Matrix) der Bögen
c	Kostenvektor
λ	Kapazitätsvektor der Ressourcen
e_k	Einheitsvektor der Dimension k
x	primale Variable
π	duale Variable
$\mathcal{DT}$	Menge der Dienstarten
$\mathcal{D}$	Menge der Dienste
Φ	Dienstelementbestandteil-Dienst-Inzidenzmatrix
$\Phi_{i\cdot}$	i-te Zeile der Matrix Φ
B	Base Constraints Matrix
ω	Kostenvektor
$\mathcal{P}$	Menge der Pfade eines Digraphen
$\mathcal{FP}$	zulässige Pfadmenge
$\mathcal{IP}$	unzulässige Pfadmenge
χ	Dienstelementbestandteil-Pfad-Inzidenzmatrix
$\chi_{\cdot P}$	P-te Spalte der Matrix χ

Literaturverzeichnis

[AMO93] Ahuja, R.K.; Magnati, T.L.; Orlin, J.B. *Network Flows: Theory, algorithms and applications*, Prentice Hall, Englewood Cliffs, NJ. (1993).

[BC89] Beasley, J.E.; Christofides, N. *An Algorithm for the Resource Constrained Shortest Path Problem*, Networks 19, 379-394, (1989).

[BGL01] Borndörfer, R.; Grötschel, M.; Löbel, A., *Scheduling Duties by Adaptive Column Generation*, ZIB-Report 01-02, Berlin, (2001).

[BLSV98] Borndörfer, R.; Löbel, A.; Strubbe, U., Völker, M., *Zielorientierte Dienstplanoptimierung*, ZIB-Preprint SC 98-41, Berlin, (1998).

[BLW02] Borndörfer, R.; Löbel, A.; Weider, S., *Integrierte Umlauf- und Dienstbildung im Öffentlichen Nahverkehr*, ZIB-Preprint SC 02-10, Berlin, (2002).

[DS92] Desrochers, M.; Soumis, F., *A Generalized Permanent Labeling Algorithm dor the Shorthest Path Problem with Time Windows*, INFOR 26pp.191-212 (1988).

[D59] Dijkstra, E., *A Note on Two Problems in Connexion with Graphs*, Numerical Mathematics 251, 215-248 (1959).

[D02] Dumitrescu, I., *Constrained Path and Cycle Problems*, Dissertation (2002).

[FHW00] Freling, R.; Huisman D.; Wagelmans, A.P.M., *Applying an integrated approach to vehicle and crew scheduling in practice*, Econometric Institute Report 194, Erasmus University Rotterdam (2000).

[GJ79] Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completness* Freeman, San Francisco (1979).

[GLS88] Grötschel, Martin; Lovász, László; Schrijver, Alexander *Geometric algorithms and combinatorial optimization. 2. corr. ed. (English)* Springer Verlag Berlin (1988).

[H92] Hassin, R. *Approximation Schemes for the Restricted Shortest Path Problem*, Mathematics of Operations Research 17, 36-42 (1992).

[HZ80] Handler, G.Y.; Zang, I. *A Dual Algorithm for the Constrained Shortest Path Problem*, Networks 10, 293-310 (1980).

[I02] ILOG S.A. *ILOG CPLEX 8.0 Reference Manual.* (2002)

[J66] Joksch, H.C. *The Shortest Route Problem with Constraints*, Journal of Mathematical Analysis and Applications 14, 191-197 (1966).

[L76] Lawler, E.I. *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Wilson, New York (1976).

[LS95] Löbel, A.; Strubbe, U. *Wagenumlaufoptimierung - Methodischer Ansatz und praktische Anwendung*, Springer. Lect. Preprint ZIB SC-95-38, Heureka 96, (1995).

[M84] Mehlhorn, K. *Data structures and algorithms*, Springer Verlag Berlin (1984).

[MZ00] Mehlhorn, K.; Ziegelmann, M. *Resource Constrained Shortest Paths*, Springer Lect. Notes Comput. Sci. 1879, 326-337 (2000).

[NW88] Nemhauser, G.; Wolsey, L. *Integer and Combinatorial Optimization*, John Wiley & Sons (1988).

[S03] Schrijver, A. *Combinatorial Optimization - Polyhedra and Effiency*, Springer Verlag Berlin (2003).

[Z01] Ziegelmann, M. *Constrained Shortest Paths and Related Problems*, Dissertation (2001).