Ein Branch&Cut-Algorithmus für das asymmetrische Travelling Salesman Problem mit Zeitfenstern - ein Vergleich verschiedener Modellierungen

Diplomarbeit
von
Sven Bogatsch

eingereicht
am Fachbereich Mathematik
der Technischen Universität Berlin
im Juli 1996

Erstgutachter: Prof. Dr. Martin Grötschel
Zweitgutachter: Prof. Dr. Rolf H. Möhring

Inhaltsverzeichnis

Einleitung

Die deutsche Wirtschaft steht vor Entwicklungen und Herausforderungen, die erhebliche wirtschaftliche Strukturveränderungen erfordern, wenn die Unternehmen im internationalen Wettbewerb mithalten und die Arbeitnehmer weiterhin hohe Löhne verdienen wollen. Besonders die **Globalisierung der Märkte** stellt die Unternehmen und die Arbeitskräfte in einen schärferen, weltweiten Wettbewerb. Moderne Verkehrs- und Kommunikationsnetze erhöhen die Mobilität der Produktionsfaktoren (Kapital, qualifizierte Arbeit, Wissen) und machen Produktionsstandorte zunehmend austauschbar.

Die Zukunftssicherung des Standortes Deutschland bedeutet heute vor allem eine klare Priorität für **Forschung und Innovation**. Eine zentrale Bedeutung hat der Technologietransfer zwischen Wissenschaft und Wirtschaft bei zunehmender Bedeutung der Kommunikations- und Informationstechnik. Schon heute ist das Hand-in-Hand-Gehen von Wirtschaft, Informatik und den naturwissenschaftlichen Bereichen, insbesondere der Mathematik, unverzichtbar geworden.

Ein Beispiel für die anwendungsbezogene Forschung und die Kooperation zwischen Wissenschaft und Wirtschaft ist das **Konrad-Zuse-Zentrum für Informationstechnik Berlin*** (ZIB). Hier wird in enger Zusammenarbeit mit den Hochschulen Forschung und Entwicklung unter anderem im Bereich der *Algorithmischen Diskreten Mathematik* betrieben. Typische Anwendungsbeispiele der Algorithmischen Diskreten Mathematik sind der *Güter- und Personenverkehr* (z.B. Fahrzeugumlaufplanung im Öffentlichen Personennahverkehr, Fahrzeugeinsatzplanung im Behindertentransport) und die *innerbetriebliche Transport,- Lager- und Umschlagslogistik* (z.B. Optimierung von Flexiblen Fertigungssystemen).

Der Schwerpunkt der Algorithmischen Diskreten Mathematik liegt auf der Untersuchung praxisrelevanter, schwieriger ("$\mathcal{NP}$-schwerer") kombinatorischer Optimierungsprobleme. Ziel ist die mathematische Analyse der hier auftretenden Modellierungen und weiterführend die Entwicklung und Implementierung effizienter Verfahren zur Lösung realer Problembeispiele. Im Mittelpunkt des Algorithmenentwurfs stehen vor allem Schnittebenenverfahren und **Branch&Cut-Algorithmen** (B&C-Algorithmen). Diese werden durch Heuristiken zur approximativen Lösung sehr großer Anwendungsbeispiele ergänzt.

Die vorliegende Arbeit entstand im Rahmen meiner studentischen Tätigkeit in der Abteilung Kombinatorische Optimierung des Konrad-Zuse-Zentrums auf Anregung von Dr. Norbert Ascheuer und Prof. Dr. Martin Grötschel. Im Mittelpunkt steht die mathematische und algorithmische Analyse verschiedener Modellierungen für das **Asymmetrische Travelling Salesman Problem mit Zeitfenstern** (ATSPTW). Zur algorithmischen Analyse verwende ich einen B&C-

*Siehe World Wide Web (WWW) unter "http://www.zib-berlin.de".

Algorithmus, der alternativ mit drei verschiedenen Modellierungen des ATSPTW arbeiten kann. Dem ATSPTW liegt das wohl bekannteste Problem der Kombinatorischen Optimierung, das Travelling Salesman Problem (TSP), zugrunde. Das TSP besteht darin, eine kürzeste Rundreise durch n gegebene Städte zu finden. Ist dabei die Entfernung von A nach B ungleich der Entfernung von B nach A, so reden wir von einem asymmetrischen TSP (ATSP). Sollen zusätzlich die Städte innerhalb eines gegebenen Zeitraumes (Zeitfenster) besucht werden, so erhalten wir ein ATSPTW. In der Praxis treten diese Probleme in sehr vielfältiger Form vor allem als Routing und Scheduling Probleme auf.

Die vorliegende Arbeit untergliedert sich in sieben Kapitel. Das erste Kapitel ist ein Ausflug in die Graphentheorie und beinhaltet vor allem die in dieser Arbeit verwendeten graphentheoretischen Notationen. Die beiden nachfolgenden Kapitel haben einführenden Charakter. Hier gehe ich auf kombinatorische Optimierungsprobleme in allgemeiner Form ein und erkläre die Grundidee von B&C-Algorithmen. In dem Kapitel 4 stelle ich, ausgehend von der mathematischen Modellierung des klassischen ATSP, drei verschiedene Modellierungen für das ATSPTW vor, die mathematisch und algorithmisch analysiert werden sollen. Darüber hinaus möchte ich die vorliegende Arbeit durch ein Beispiel aus der Praxis (Optimierung Flexibler Fertigungssysteme) motivieren. Der entscheidende Teil eines B&C-Algorithmus ist die Berechnung der unteren Schranken mit Hilfe des Schnittebenenverfahrens. Hierbei ist die Kenntnis problemspezifischer Schnittebenen von großer Bedeutung. Im Kapitel 5 gebe ich einen Überblick über verschiedene Schnittebenen, die für das ATSPTW verwendet werden können. Es werden Separierungsalgorithmen vorgestellt, die zur Untersuchung des Einflusses verschiedener Klassen von Ungleichungen auf den B&C-Algorithmus von mir implementiert worden sind. Der zur algorithmischen Analyse verwendete B&C-Algorithmus wird im Kapitel 6 beschrieben. Dabei werden auch die Details der Implementierung des Algorithmus erklärt. Im letzten Kapitel werden dann die numerischen Ergebnisse des B&C-Algorithmus für die verschiedenen Modellierungen des ATSPTW präsentiert und analysiert. Neben dem Vergleich der verschiedenen mathematischen Modellierungen werden die Einflüsse der in dem B&C-Algorithmus neu verwendeten Klassen von Ungleichungen numerisch analysiert und ausgewertet.

1 Graphentheoretische Grundbegriffe

Viele Probleme der Kombinatorischen Optimierung werden mit Begriffen und Symbolen aus der Graphentheorie beschrieben. Aus diesem Grund seien die in dieser Arbeit verwendeten Grundbegriffe der Graphentheorie den weiteren Betrachtungen vorangestellt. Für einen umfassenderen Überblick verweise ich auf Jungnickel [Jun90].

Ein **Graph** G ist ein Paar (V, E), bestehend aus einer endlichen **Knotenmenge** $V \neq \emptyset$ und einer **Kantenmenge** E von zweielementigen Teilmengen von V. Die Elemente aus V heißen **Knoten**, die aus E **Kanten**. Zu jeder Kante $e \in E$ gibt es also Knoten $u, v \in V$ mit $e = uv = vu$. Wir sagen, die Knoten u und v sind mit der Kante e **inzident**, und Knoten u ist mit Knoten v **adjazent**.

Ein gerichteter Graph, d.h. ein Graph, dessen Kanten eine Orientierung haben, bezeichnen wir als **Digraph** $D = (V, A)$. Er besteht aus der endlichen Knotenmenge $V \neq \emptyset$ und einer **Bogenmenge** A. Ein **Bogen** $a \in A$ ist ein geordnetes Paar von Knoten, d.h. $a = (u, v)$. Der Knoten u heißt **Anfangsknoten**, der Knoten v **Endknoten** des Bogens a.

Seien $D_1 = (V_1, A_1)$ und $D_2 = (V_2, A_2)$ zwei Digraphen, so daß $V_2 \subseteq V_1$ und $A_2 \subseteq A_1$, dann heißt der Digraph D_2 **Unterdigraph** von D_1.

Ein Digraph $D = (V, A)$ heißt **vollständig**, wenn zu je zwei Knoten $u, v \in V$, $u \neq v$, die Bögen (u, v) und (v, u) in A enthalten sind. Ein Digraph heißt **transitiv abgeschlossen**, wenn für paarweise voneinander verschiedene Knoten $u, v, w \in V$ aus $(u, v) \in A$ und $(v, w) \in A$ immer $(u, w) \in A$ folgt.

Für einen Digraphen $D = (V, A)$ und eine Knotenmenge $W \subseteq V$ bezeichnet

$$A(W) := \{(u, v) \in A | u, v \in W\}$$

die Menge der Bögen mit Anfangs- und Endknoten in W. Seien W_1 und W_2 zwei nichtleere Teilmengen von V mit $W_1 \cap W_2 = \emptyset$, dann bezeichnet

$$(W_1 : W_2) := \{(u, v) \in A | u \in W_1, v \in W_2\}$$

die Bogenmenge mit Anfangsknoten in W_1 und Endknoten in W_2. Für $W \subset V$, $W \neq \emptyset$, sei

$$\delta^+(W) := \{(u, v) \in A | u \in W, v \notin W\}$$

die Menge aller Bögen mit Anfangsknoten in W und Endknoten in $V \backslash W$ und

$$\delta^-(W) := \{(u, v) \in A | u \notin W, v \in W\}$$

die Menge aller Bögen mit Endknoten in W und Anfangsknoten in $V \backslash W$. Offensichtlich gilt $\delta^+(W) = \delta^-(V \backslash W)$. Die Bogenmenge $\delta^+(W)$ (bzw. $\delta^-(W)$) heißt **Schnitt**. Für $\delta^+(\{v\})$ bzw. für $\delta^-(\{v\})$ schreiben wir kurz $\delta^+(v)$ bzw. $\delta^-(v)$.

In einem Digraphen heißt eine endliche Folge K von alternierend auftretenden Knoten und Bögen, d.h. $K = (v_0, a_1, v_1, a_2, v_2, \ldots, a_k, v_k), k \geq 0$, eine **Kette**, falls jeder Bogen a_i mit den Knoten v_{i-1} und v_i inzidiert. Die Zahl k heißt **Länge** der Kette. Der Knoten v_0 ist der **Anfangsknoten**, v_k der **Endknoten** der Kette. Die Knoten $v_1, \ldots, v_{k-1}$ heißen **innere Knoten** der Kette. Gilt $a_i = (v_{i-1}, v_i)$ für $i = 1, \ldots, k$, so ist K eine **gerichtete Kette**. Eine (gerichtete) Kette, in der alle Knoten voneinander verschieden sind, heißt (gerichteter) **Weg**. Wir beschreiben einen gerichteten Weg P der Länge k im folgenden durch die Bogenmenge

$$P = \{(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\}.$$

Zur Vereinfachung der Notation schreiben wir auch

$$P = (v_0, \ldots, v_k).$$

Ein (gerichteter) Weg der Länge $|V| - 1$ heißt (gerichteter) **Hamiltonweg**. Eine Kette ist **geschlossen**, falls $k \neq 0$ und $v_0 = v_k$. Eine geschlossene (gerichtete) Kette, in der der Anfangsknoten und alle inneren Knoten voneinander verschieden sind, heißt (gerichteter) **Kreis**. Einen gerichteten Kreis C beschreiben wir im folgenden durch die Bogenmenge

$$C = \{(v_0, v_1), (v_1, v_2), \ldots, (v_{k-2}, v_{k-1}), (v_{k-1}, v_0)\}.$$

Zur Vereinfachung der Notation schreiben wir auch

$$C = (v_0, v_1, \ldots, v_{k-1}, v_0).$$

Die Bogenmenge C beschreibt also einen gerichteten Kreis, der im Knoten v_0 beginnt und endet. Ein (gerichteter) Kreis der Länge $|V|$ heißt (gerichteter) **Hamiltonkreis** bzw. (gerichtete) **Tour**.

Ein Digraph $D = (V, A)$ heißt **azyklisch**, falls er keinen gerichteten Kreis enthält. Ein Digraph, der einen gerichteten Hamiltonkreis enthält, heißt **hamiltonsch**.

Sind in einem Digraphen $D = (V, A)$ Gewichte c_a für alle $a \in A$ gegeben, so reden wir von einem **gewichteten** Digraphen. Das **Gewicht** einer Bogenmenge $B \subseteq A$ ist dann durch

$$c(B) = \sum_{a \in B} c_a$$

gegeben.

2 Grundlegende Konzepte der Kombinatorischen Optimierung

In diesem Kapitel möchte ich den mathematischen Hintergrund für die vorliegende Arbeit darstellen. Im allgemeinen werde ich mich dabei auf Grötschel [Grö94] beziehen. Im ersten Abschnitt möchte ich einen kurzen Einblick in die Komplexitätstheorie geben, welche für die Kombinatorische Optimierung von Bedeutung ist. Für eine formale Einführung in dieses Gebiet verweise ich auf Garey und Johnson [GJ79]. Im zweiten Abschnitt werden kombinatorische Optimierungsprobleme in allgemeiner Form eingeführt. Desweiteren soll eine Beziehung zwischen kombinatorischen und linearen Optimierungsproblemen hergestellt werden. Hierbei habe ich mich auf die Arbeit von Grötschel und Padberg [GP85] gestützt. Kenntnisse aus der Linearen Optimierung werden vorausgesetzt.

Als wichtiges Teilgebiet der Kombinatorischen Optimierung hat sich die **Polyederische Kombinatorik** herausgestellt. Sie kann als Synthese von Polyedertheorie und Kombinatorischer Optimierung aufgefaßt werden. Polyedertheoretische Ansätze zur Lösung kombinatorischer Optimierungsprobleme erwiesen sich mit dem Ende der siebziger Jahre zunehmend als sehr erfolgreich, wie es z.B. die Arbeit von Grötschel [Grö80] zeigt, in der ein Travelling Salesman Problem (TSP) mit 120 Städten mittels eines polyedertheoretischen Ansatzes gelöst wurde. Das rasante Ansteigen der Rechnerleistungen und die Entwicklungen in der Polyederischen Kombinatorik lassen heutzutage die Lösung weitaus größerer Probleme zu (z.B das 4461-Städte-TSP von Applegate, Bixby, Chvàtal und Cook (1993)). Eine gute Einführung in die Polyedertheorie bietet die schon erwähnte Arbeit von Grötschel und Padberg [GP85].

Im folgenden bezeichnen wir mit $\mathbb{N}$ die Menge der natürlichen Zahlen. Desweiteren bezeichnen wir mit $\mathbb{R}$ den Körper der reellen Zahlen, mit $\mathbb{Q}$ den Körper der rationalen Zahlen und mit $\mathbb{K}$ einen beliebigen dieser beiden angeordneten Körper.

2.1 Die Klassen $\mathcal{P}$ und $\mathcal{NP}$, $\mathcal{NP}$-Vollständigkeit

Kombinatorische Optimierungsprobleme lassen sich aus komplexitätstheoretischer Sicht in "einfache" und "schwierige" Probleme unterteilen. Zunächst benötigen wir jedoch den Begriff eines polynomialen Algorithmus. Ist A ein Algorithmus zur Lösung eines gegebenen Problems Π, so definieren wir dessen **Laufzeit** als die Anzahl der Elementaroperationen (Addition, Multiplikation, etc.), welche auf einem Computer zur Lösung eines Problembeispiels $\mathcal{I} \in \Pi$ ausgeführt werden müssen. Die Laufzeit eines Algorithmus A wird durch die Funktion $f_A : \mathbb{N} \to \mathbb{N}$ ausgedrückt. Sie gibt für jede natürliche Zahl n die Anzahl $f_A(n)$ der Elementaroperationen an, die der Algorithmus A zur Lösung eines Problembeispiels $\mathcal{I}$

der Größe n höchstens (worst case) benötigt. Zur Beschreibung der Laufzeit eines Algorithmus wird auch oft die O-Notation verwendet.

Definition 2.1
Seien $f : \mathbb{N} \to \mathbb{N}$ und $g : \mathbb{N} \to \mathbb{N}$ zwei natürliche Funktionen. Wir schreiben $f(n) = O(g(n))$ für $n \in \mathbb{N}$ genau dann, wenn zwei positive Konstanten c und n_0 existieren, so daß $f(n) \leq c \cdot g(n)$ für alle $n \geq n_0$ gilt.

Wir nennen nun einen Algorithmus A **polynomial**, wenn die Laufzeit von A zur Lösung eines Problems Π durch ein Polynom beschränkt ist, d.h $f_A = O(n^k)$, wobei k der Grad des Polynoms ist. Eine exponentielle Laufzeit wäre z.B. $O(2^n)$.

Betrachten wir nun zunächst nur Entscheidungsprobleme, also Probleme, dessen Lösung eine Frage mit "ja" oder "nein" beantwortet. Ein Entscheidungsproblem Π gehört zur Klasse $\mathcal{P}$, wenn zu dessen Lösung ein polynomialer Algorithmus existiert. Ein Entscheidungsproblem gehört zur Klasse $\mathcal{NP}$, wenn für jedes Problembeispiel $\mathcal{I} \in \Pi$, für das die Antwort "ja" lautet, die Korrektheit der "ja"-Antwort in polynomialer Zeit verifiziert werden kann. Offensichtlich gilt $\mathcal{P} \subset \mathcal{NP}$. Dagegen ist die Frage, ob $\mathcal{P} = \mathcal{NP}$ gilt oder nicht, immer noch ungelöst. Sie gilt als herausragendes offenes Problem der Komplexitätstheorie.

Die schwierigsten Probleme in der Klasse $\mathcal{NP}$ sind die sogenannten $\mathcal{NP}$-vollständigen Entscheidungsprobleme. Ein Entscheidungsproblem Π heißt **$\mathcal{NP}$-vollständig**, falls es in $\mathcal{NP}$ liegt, und falls jedes andere Problem aus $\mathcal{NP}$ in polynomialer Zeit in Π transformiert werden kann. Somit folgt aus der polynomialen Lösbarkeit eines $\mathcal{NP}$-vollständigen Problems die polynomiale Lösbarkeit aller Probleme in $\mathcal{NP}$. Gelingt es also, ein $\mathcal{NP}$-vollständiges Problem in polynomialer Zeit zu lösen, so würde $\mathcal{P} = \mathcal{NP}$ folgen.

Nun gibt es zu jedem Optimierungsproblem Π ein zugehöriges Entscheidungsproblem. Sei Π ein gegebenes Minimierungsproblem (bzw. Maximierungsproblem). Dann legt man zusätzlich zu jedem Problembeispiel $\mathcal{I} \in \Pi$ eine Schranke S fest und fragt, ob es eine Lösung des Problembeispiels $\mathcal{I}$ gibt, dessen Wert nicht größer (bzw. nicht kleiner) als S ist. Ein Optimierungsproblem nennen wir nun **$\mathcal{NP}$-schwer**, falls das zugehörige Entscheidungsproblem $\mathcal{NP}$-vollständig ist. Zu den $\mathcal{NP}$-schweren Optimierungsproblemen gehören z.B. das schon erwähnte TSP und das asymmetrische Travelling Salesman Problem (ATSP).

2.2 Kombinatorische Optimierungsprobleme

In diesem Abschnitt werden wir kombinatorische Optimierungsprobleme in einer allgemeinen Form vorstellen und die Herangehensweise zur Lösung derartiger Probleme beschreiben. Ein kombinatorisches Optimierungsproblem läßt sich wie folgt formulieren.

Kombinatorisches Optimierungsproblem. *Gegeben seien eine endliche Menge E, eine Teilmenge $\mathcal{I}$ der Potenzmenge 2^E von E und eine Funktion $c : E \rightarrow \mathbb{K}$. Für jede Menge $F \subseteq E$ definieren wir ihren "Wert" durch $c(F) := \sum_{e \in F} c(e)$. Gesucht ist eine Menge $I^* \in \mathcal{I}$, so daß $c(I^*)$ das Minimum (bzw. das Maximum) für alle $I \in \mathcal{I}$ ist.*

Ein solches kombinatorisches Optimierungsproblem bezeichnen wir im weiteren mit $(E, \mathcal{I}, c)$. Die Menge E heißt **Grundmenge**, c ist die **Zielfunktion**, und die Menge $\mathcal{I}$ ist die Menge aller **zulässigen Lösungen**, d.h., jedes Element von $\mathcal{I}$ ist eine zulässige Lösung von $(E, \mathcal{I}, c)$. Die Menge $\mathcal{I}$ wird durch

$$\mathcal{I} := \{I \subseteq E \mid I \text{ hat Eigenschaft } \Pi\}$$

definiert, wobei die Eigenschaft Π problemabhängig ist.

Da die Menge der zulässigen Lösungen $\mathcal{I}$ endlich ist, könnte man im Prinzip kombinatorische Optimierungsprobleme durch vollständige Enumeration lösen. Für viele kombinatorische Optimierungsprobleme wächst jedoch die Anzahl der Elemente in $\mathcal{I}$ exponentiell mit der Anzahl der Elemente in der Grundmenge E. Ist z.B. die Menge der zulässigen Lösungen für das ATSP durch die Menge aller gerichteten Touren in einem vollständigen Digraphen auf n Knoten gegeben, so ist $|E| = n(n-1)$ und $|\mathcal{I}| = (n-1)!$. In einem solchen Fall ist eine vollständige Enumeration aller zulässigen Lösungen praktisch hoffnungslos. Schon für $n = 15$ gibt es für das ATSP 87.178.291.200 zulässige Lösungen. Hier müssen andere Lösungsmethoden gefunden werden. Eine bekannte Methode ist das sogenannte Branch&Bound–Verfahren, welches durch implizites Enumerieren mit vertretbarem Aufwand eine Optimallösung finden kann. Eine genauere Beschreibung dieses Verfahrens wird in Kapitel 3 gegeben.

Bevor wir eine Formulierung eines kombinatorischen Optimierungsproblems als lineares Programm angeben, benötigen wir noch einige Begriffe. Eine Teilmenge $G \subseteq \mathbb{K}^n$ heißt **Hyperebene**, falls es einen Vektor $a \in \mathbb{K}^n \setminus \{0\}$ und einen Skalar $a_0 \in \mathbb{K}$ gibt, so daß $G = \{x \in \mathbb{K}^n \mid a^T x = a_0\}$. Eine Teilmenge $H \subseteq \mathbb{K}^n$ heißt **Halbraum**, falls es einen Vektor $a \in \mathbb{K}^n \setminus \{0\}$ und einen Skalar $a_0 \in \mathbb{K}$ gibt, so daß $H = \{x \in \mathbb{K}^n \mid a^T x \leq a_0\}$. Ein **Polyeder** ist der Durchschnitt endlich vieler Halbräume, d.h., jedes Polyeder P kann in der Form $P = \{x \in \mathbb{K}^n \mid Ax \leq b\}$ dargestellt werden. Für eine Menge von Vektoren $S = \{v_1, \ldots, v_k\}$ in $\mathbb{K}^n$ heißt $\operatorname{conv}(S)$ die **konvexe Hülle** von S, d.h. die Menge aller Vektoren $x \in \mathbb{K}^n$, die sich durch $x = \lambda_1 v_1 + \ldots + \lambda_k v_k$ darstellen lassen, wobei $\lambda_1, \ldots, \lambda_k$ nichtnegative Zahlen mit $\sum_{i=1}^k \lambda_i = 1$ sind. Jede Menge, die als konvexe Hülle endlich vieler Vektoren dargestellt werden kann, heißt **Polytop**.

Sei E eine endliche Menge, so bezeichnen wir mit $\mathbb{K}^E$ den $\mathbb{K}$–Vektorraum der $|E|$–Tupel $x = (x_e)_{e \in E}$, so daß jede Komponente von x mit einem Element von E indiziert ist. Somit haben wir eine eineindeutige Zuordnung eines Elements $e \in E$

zu einer Komponente x_e von x. Ist $F \subseteq E$, so heißt der Vektor $\chi^F \in \mathbb{K}^E$ mit

$$\chi_e^F = \begin{cases} 1 & \text{falls } e \in F \\ 0 & \text{falls } e \notin F \end{cases}$$

der **Inzidenzvektor** von F. Für ein gegebenes kombinatorisches Optimierungsproblem $(E, \mathcal{I}, c)$ definieren wir nun durch

$$P_{\mathcal{I}} := \text{conv}\{\chi^I \in \mathbb{K}^E \mid I \in \mathcal{I}\} \tag{2.1}$$

die konvexe Hülle der Inzidenzvektoren der zulässigen Lösungen $I \in \mathcal{I}$. Wir gehen davon aus, daß die Menge der zulässigen Lösungen $\mathcal{I}$ nicht leer, aber endlich ist. Somit ist $P_{\mathcal{I}}$ als konvexe Hülle über endlich viele Vektoren ein konvexes Polytop. Die Ecken von $P_{\mathcal{I}}$ sind genau die Inzidenzvektoren der zulässigen Lösungen aus der Menge $\mathcal{I}$.

Im folgenden betrachten wir o.B.d.A. nur noch Minimierungsprobleme. Ein Maximierungsproblem kann durch Multiplikation der Zielfunktion mit -1 auf ein Minimierungsproblem transformiert werden. Fassen wir die Funktion $c : E \to \mathbb{K}$ als Vektor $c \in \mathbb{K}^E$ auf, so beschreibt

$$\begin{array}{ll} \min & c^T x \\ & x \in P_{\mathcal{I}} \end{array} \tag{2.2}$$

ein lineares Programm. Jede optimale Ecklösung dieses linearen Programms entspricht dann einem Inzidenzvektor einer Optimallösung von $(E, \mathcal{I}, c)$ und umgekehrt. Wollen wir das lineare Programm (2.2) jedoch algorithmisch lösen, benötigen wir eine Beschreibung des Polytops $P_{\mathcal{I}}$ (2.1) durch lineare Gleichungen und Ungleichungen, d.h. eine Darstellung der Form

$$P_{\mathcal{I}} = \{x \in \mathbb{K}^E \mid Bx = d, Ax \leq b\}. \tag{2.3}$$

Läßt sich das Polytop $P_{\mathcal{I}}$ wie in (2.3) darstellen, so heißt das System von Gleichungen und Ungleichungen $Bx = d, Ax \leq b$ eine **vollständige Beschreibung** von $P_{\mathcal{I}}$. Wir nennen das System von Gleichungen und Ungleichungen **nicht redundant**, falls keine der Gleichungen bzw. Ungleichungen weggelassen werden kann, ohne daß sich das Polytop $P_{\mathcal{I}}$ verändert. Ziel der Polyederischen Kombinatorik ist es, ein nicht redundantes System von Gleichungen und Ungleichungen zu finden, welches $P_{\mathcal{I}}$ vollständig beschreibt. Nach einem Resultat von Weyl (1933) läßt sich jedes Polytop theoretisch mit Hilfe von endlich vielen Ungleichungen beschreiben. Jedoch sind für bestimmte Optimierungsprobleme exponentiell viele Ungleichungen zur Beschreibung von $P_{\mathcal{I}}$ (2.1) notwendig, so daß der naheliegende Ansatz, alle bekannten Ungleichungen aufzulisten und Methoden der Linearen Optimierung zur Lösung des Optimierungsproblems anzuwenden, nicht durchführbar ist. Darüber hinaus sind für viele $\mathcal{NP}$-schwere Optimierungsprobleme $(E, \mathcal{I}, c)$ derzeit noch keine vollständigen Beschreibungen von $P_{\mathcal{I}}$ (2.1) gefunden worden.

Relativ einfach kann dagegen ein kombinatorisches Optimierungsproblem als ganzzahliges lineares Programm (englisch: integer linear program, kurz: ILP), als gemischt-ganzzahliges Programm (englisch: mixed integer program, kurz: MIP) oder als binäres lineares Programm (englisch: binary linear program, kurz: BLP) formuliert werden. Binäre lineare Programme sind meist in der Form

$$\begin{array}{lrcl} & \min & c^T x & \\ (1) & Bx & = & d \\ (2) & Ax & \leq & b \\ (3) & x & \in & \{0,1\}^E \end{array} \tag{2.4}$$

gegeben. Das Polytop $P_{\mathcal{I}}$ (2.1) kann dann durch

$$P_{\mathcal{I}} = \operatorname{conv}\{x \in \mathbb{K}^E \mid x \text{ erfüllt } (2.4)\,(1) - (3)\} \tag{2.5}$$

beschrieben werden. Jedoch sind ganzzahlige lineare Optimierungsprobleme wieder $\mathcal{NP}$-schwer. Eine bekannte Methode zur Lösung $\mathcal{NP}$-schwerer Optimierungsprobleme bzw. zur Bestimmung unterer Schranken für den Wert der Optimallösung ist das sogenannte Schnittebenenverfahren, welches im Kapitel 3 detaillierter beschrieben wird.

3 Branch&Cut-Algorithmen zur Lösung $\mathcal{NP}$-schwerer Optimierungsprobleme

Unter einem Algorithmus versteht man eine endliche, vollständige Vorschrift zur Lösung einer bestimmten Klasse von Problemen in einer Folge von Schritten, die in endlicher Zeit ausgeführt werden können. Von diesem einfachen Grundverständnis für Algorithmen ausgehend, möchte ich in diesem Kapitel die Grundidee von Branch&Cut-Algorithmen vorstellen. Dabei werde ich mich auf die Arbeiten von Jünger, Reinelt und Thienel [JRT94] und [JRT95] sowie auf Grötschel [Grö94] beziehen.

Ein Branch&Cut-Algorithmus ist eine spezielle Variante eines Branch&Bound-Algorithmus, bei dem die unteren Schranken durch ein Schnittebenenverfahren bestimmt werden. Die Verbindung eines Enumerationsschemas (**branch**) mit einem Schnittebenenverfahren (**cut**) geben diesen Algorithmen den Namen **branch&cut**, welcher von Padberg und Rinaldi [PR91] eingeführt wurde.

Um die Bedeutung der Schnittebenenverfahren hervorzuheben, möchte ich im ersten Abschnitt dieses Kapitels zunächst auf diese Verfahren in ihrer reinen Form eingehen, bevor ich im zweiten Abschnitt die Grundidee von Branch&Cut-Algorithmen skizzieren werde.

3.1 Schnittebenenverfahren

Wie wir im Kapitel 2 gesehen haben, läßt sich für viele kombinatorische Optimierungsprobleme $(E, \mathcal{I}, c)$ das Polytop $P_\mathcal{I}$ (2.1) durch

$$P_\mathcal{I} = \operatorname{conv}\{x \in \mathbb{K}^E \mid Bx = d, Ax \leq b, x \in \{0,1\}^E\} \tag{3.1}$$

darstellen, wobei jede Ecke von $P_\mathcal{I}$ nur aus den Komponenten 0 oder 1 besteht. Mit dieser Darstellung erhalten wir durch

$$\begin{array}{ll} \min & c^T x \\ & x \in P_\mathcal{I} \end{array} \tag{3.2}$$

eine Formulierung von $(E, \mathcal{I}, c)$ als ganzzahliges Programm. Um $(E, \mathcal{I}, c)$ mit Hilfe linearer Optimierungsmethoden (z.B. Simplexverfahren oder Ellipsoidmethode) zu lösen, benötigen wir eine geeignete Relaxierung von $(E, \mathcal{I}, c)$.

Definition 3.1
Gegeben sei ein kombinatorisches Optimierungsproblem $(E, \mathcal{I}, c)$. Jedes lineare Programm $\min\{c^T x \mid Ax \leq b, x \geq 0\}$ mit der Eigenschaft, daß $\{x \in \mathbb{K}^E \mid Ax \leq b, x \geq 0, x$ ganzzahlig$\}$ die Menge der Inzidenzvektoren von $\mathcal{I}$ ist, heißt **LP-Relaxierung** *von $(E, \mathcal{I}, c)$.*

Bei LP-Relaxierungen sollte das Polyeder $\{x \in \mathbb{K}^E \mid Ax \leq b, x \geq 0\}$ möglichst "nahe" an $\text{conv}\{\chi^I \in \mathbb{K}^E \mid I \in \mathcal{I}\}$ liegen.

Definieren wir nun das Polytop P durch

$$P = \{x \in \mathbb{K}^E \mid Bx = d, Ax \leq b, 0 \leq x \leq 1\}, \tag{3.3}$$

so ist

$$\begin{array}{ll} \min & c^T x \\ & x \in P \end{array} \tag{3.4}$$

eine LP-Relaxierung von $(E, \mathcal{I}, c)$. Diese Relaxierung ist eine "Einbettung" des Problems (3.2) in das größere Problem (3.4), d.h., es gilt

$$P_{\mathcal{I}} \subseteq P \tag{3.5}$$

und damit auch

$$\min\{c^T x \mid x \in P\} \leq \min\{c^T x \mid x \in P_{\mathcal{I}}\}. \tag{3.6}$$

Darüber hinaus ist jede Ecke von $P_{\mathcal{I}}$ auch eine Ecke von P. Dies bedeutet, daß eine ganzzahlige Optimallösung x^* der LP-Relaxierung (3.4) auch eine Optimallösung von $(E, \mathcal{I}, c)$ ist. In diesem Fall ist also $x^* \in P_{\mathcal{I}}$, und wir hätten $(E, \mathcal{I}, c)$ gelöst. Falls $x^* \notin P_{\mathcal{I}}$, müssen wir bezüglich $P_{\mathcal{I}}$ gültige Ungleichungen finden, die von x^* verletzt werden. Derartige Ungleichungen werden wir auch als **Schnittebenen** bezeichnen.

Definition 3.2
Es sei min$\{c^T x \mid Ax \leq b, x \text{ ganzzahlig}\}$ *eine ILP-Formulierung eines kombinatorischen Optimierungsproblems* $(E, \mathcal{I}, c)$ *und* $a \in \mathbb{K}^E, a_0 \in \mathbb{K}$. *Eine Ungleichung* $a^T x \leq a_0$ *heißt* **gültig** *bzgl.* $P_{\mathcal{I}} = conv\{x \in \mathbb{K}^E \mid Ax \leq b, x \text{ ganzzahlig}\}$, *falls* $a^T \overline{x} \leq a_0$ *für alle zulässigen Lösungen* $\overline{x} \in P_{\mathcal{I}}$ *gilt.*

Von besonderer Bedeutung sind gültige Ungleichungen, die einen "tiefen" Schnitt in das Polyeder P (3.3) versprechen. Sei $Q \subseteq \mathbb{K}^n$ ein Polyeder. Eine **Seitenfläche** von Q ist eine Teilmenge S von Q, für die eine bzgl. Q gültige Ungleichung $a^T x \leq a_0$ existiert, so daß $S = \{x \in Q \mid a^T x = a_0\}$. Eine **Facette** F von Q ist eine echte, nichtleere Seitenfläche von Q (d.h. $\emptyset \neq F \neq Q$), die maximal bzgl. Mengeninklusion ist. Für eine detailliertere Einführung in diesen Teil der Polyedertheorie siehe Grötschel und Padberg [GP85]. Gültige Ungleichungen, die Facetten von $P_{\mathcal{I}}$ definieren, sind von besonderem Interesse, da sie in einem Schnittebenenverfahren den "tiefsten Schnitt" in das Polyeder P (3.3) versprechen.

Für viele kombinatorische Optimierungsprobleme, wie z.B. auch für das ATSP, sind große Klassen von Ungleichungen zur Beschreibung von $P_{\mathcal{I}}$ (2.1) bekannt.

Jedoch haben viele von ihnen exponentielle Größenordnung, so daß wir nicht die Lösung x^* der LP-Relaxierung (3.4) einfach in alle Ungleichungen einsetzen und überprüfen können, ob x^* die Ungleichungen erfüllt. Diese Vorgehensweise würde zu einem Algorithmus mit exponentieller Laufzeit führen. Statt dessen müssen wir das folgende Problem lösen.

Separierungsproblem. *Gegeben seien eine Klasse $Ax \leq b$ von gültigen Ungleichungen für ein kombinatorisches Optimierungsproblem $(E, \mathcal{I}, c)$ und ein Vektor $y \in \mathbb{K}^E$. Entscheide, ob y alle Ungleichungen in $Ax \leq b$ erfüllt. Falls nicht, bestimme eine bzw. mehrere Ungleichungen $a^T x \leq a_0$ aus $Ax \leq b$ mit $a^T y > a_0$.*

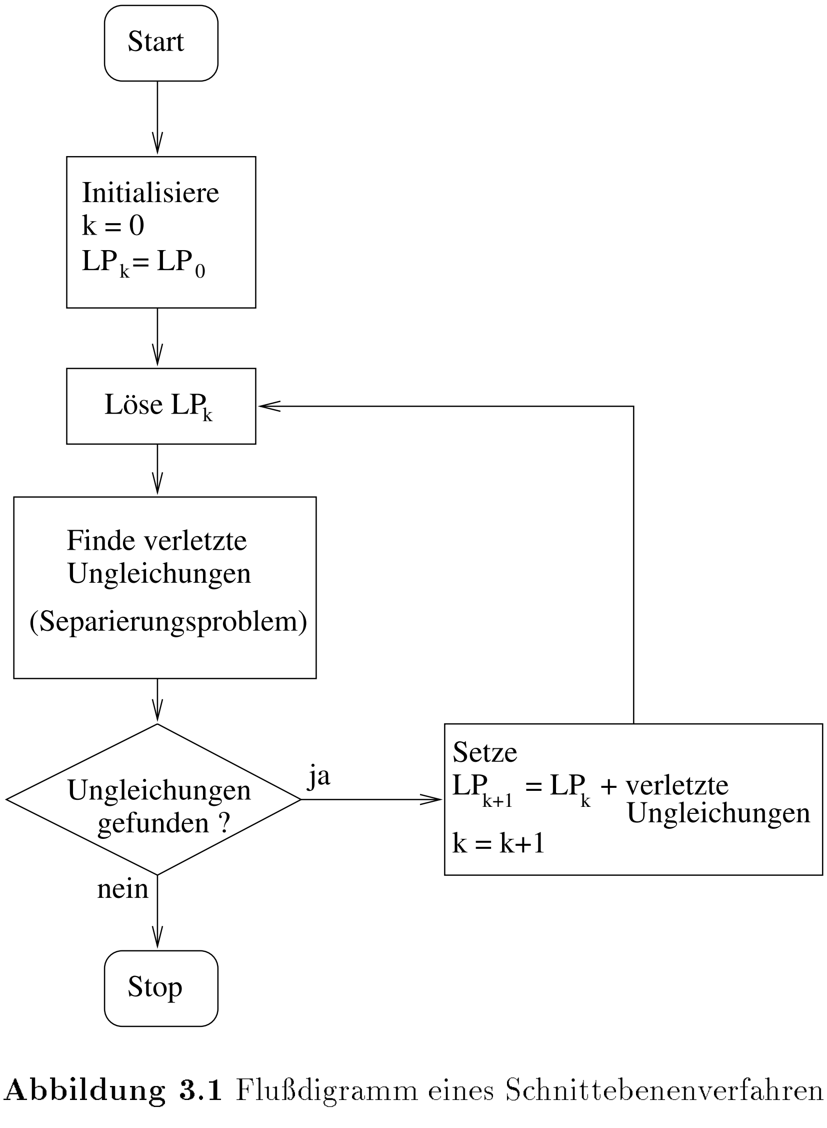

Abbildung 3.1 Flußdigramm eines Schnittebenenverfahrens

Man beachte die polynomiale Äquivalenz von Separierung und Optimierung, die von Grötschel, Lovász und Schrijver [GLS88] bewiesen worden ist. Sie besagt, daß das Optimierungsproblem $(E, \mathcal{I}, c)$ genau dann in polynomialer Zeit gelöst werden kann, wenn das Separierungsproblem für $P_{\mathcal{I}}$ in polynomialer Zeit gelöst werden kann.

Angenommen, wir haben eine Klasse von gültigen Ungleichungen für $(E, \mathcal{I}, c)$ gegeben. Ein Algorithmus, der für einen gegebenen Punkt $y \in \mathbb{K}^E$ eine oder mehrere verletzte Ungleichungen findet, heißt **Separierungsalgorithmus**. Wir nennen einen Separierungsalgorithmus **exakt**, wenn er entweder verletzte Ungleichungen findet oder die Antwort liefert, daß es keine verletzten Ungleichungen gibt. Exakte Separierungsalgorithmen sind bzgl. ihrer Laufzeit oft sehr langsam oder für bestimmte Klassen von Ungleichungen gar nicht bekannt. Darüber hinaus kann das Separierungsproblem für bestimmte Klassen von Ungleichungen sogar $\mathcal{NP}$-schwer sein. In solchen Fällen müssen wir auf einen **heuristischen** Separierungsalgorithmus zurückgreifen, der in kurzer Zeit verletzte Ungleichungen finden kann oder, ohne verletzte Ungleichungen zu finden, abbricht, wobei nicht garantiert ist, daß keine verletzten Ungleichungen existieren.

Vor diesem Hintergrund sind wir nun in der Lage, das Schnittebenenverfahren in seiner grundlegenden Version anzugeben. Die Abbildung 3.1 zeigt ein Flußdiagramm eines Schnittebenenverfahrens. Wir beginnen mit dem Anfangs-LP LP_0, das häufig einer LP-Relaxierung von $(E, \mathcal{I}, c)$ entspricht. In jedem Iterationsschritt k lösen wir das lineare Programm LP_k. Erhalten wir eine Lösung x^* für die $x^* \in P_{\mathcal{I}}$ gilt, so können wir den Algorithmus abbrechen und haben eine Optimallösung von $(E, \mathcal{I}, c)$ gefunden. Ansonsten versuchen wir, mittels exakter bzw. heuristischer Separierungsalgorithmen verletzte Ungleichungen zu finden. Falls verletzte Ungleichungen gefunden worden sind, fügen wir diese dem linearen Programm LP_k hinzu und erhalten das lineare Programm LP_{k+1}. Andernfalls terminiert der Algorithmus, und wir geben die aktuelle LP-Lösung als untere Schranke für den Wert der Optimallösung von $(E, \mathcal{I}, c)$ aus.

Wird das Schnittebenenverfahren zur Bestimmung der unteren Schranken in einem Branch&Bound-Algorithmus eingebettet, so reden wir von einem Branch-&Cut-Algorithmus.

3.2 Branch&Cut-Algorithmen

Kombinatorische Optimierungsprobleme könnten theoretisch durch vollständige Enumeration gelöst werden. Wie schon erwähnt, wächst jedoch für viele kombinatorische Optimierungsprobleme $(E, \mathcal{I}, c)$ die Anzahl der Lösungen exponentiell mit der Anzahl $n = |E|$ der Elemente der Grundmenge E. Die naheliegende Idee der vollständigen Enumeration aller Lösungen eines kombinatorischen Optimierungsproblems ist somit aus Gründen der Effizienz praktisch unbrauchbar. Man bedient sich anstelle der vollständigen Enumeration impliziter Enumerationsverfahren, sogenannten Branch&Bound-Algorithmen (B&B-Algorithmen). Diese

Methode wurde von Land und Doig (1960) und Dakin (1965) zur Lösung von gemischt-ganzzahligen Optimierungsproblemen entwickelt (siehe [Sch89]).

Padberg und Rinaldi haben sich in ihrer Arbeit [PR91] mit den Nachteilen der klassischen Herangehensweise zur Lösung $\mathcal{NP}$-schwerer kombinatorischer Optimierungsprobleme, die Schnittebenenverfahren und B&B-Algorithmen getrennt voneinander benutzt, auseinandergesetzt. Um diese Nachteile zu beseitigen, haben sie das Schnittebenenverfahren zur Bestimmung der unteren Schranken in ein B&B-Algorithmus eingebettet und nannten diese neue Version eines B&B-Algorithmus **Branch&Cut-Algorithmus** (B&C-Algorithmus). Wir werden nun im folgenden die Grundidee von B&C-Algorithmen erläutern. Für tiefergehende Details verweisen wir auf das Kapitel 6 bzw. auf [JRT95].

Die Idee von B&C-Algorithmen liegt wie bei herkömmlichen B&B-Algorithmen in dem "devide-and-conquer"-Prinzip. Wir unterteilen das Originalproblem in kleinere Teilprobleme (**branching**), während diese wiederum in weitere Teilprobleme unterteilt werden. Die einfachste Branching-Regel besteht darin, jeweils zwei neue Teilprobleme zu generieren, wobei in einem Teilproblem ein bestimmtes Element aus der Grundmenge für die Lösung verboten wird, während es für das andere Teilproblem in die Lösung gezwungen wird. Die mit diesem Element korrespondierende Variable nennen wir **Branching-Variable**.

Die Menge der Teilprobleme kann in einem binären Entscheidungsbaum repräsentiert werden, den wir **Branch&Cut-Baum** nennen. Jedes Teilproblem korrespondiert mit genau einem **Branch&Cut-Knoten** des Branch&Cut-Baumes. Die Wurzel des Branch&Cut-Baumes korrespondiert mit dem Originalproblem. Den Branch&Cut-Knoten, der gerade bearbeitet wird, nennen wir **aktuellen** Branch&Cut-Knoten. Die Branch&Cut-Knoten, die noch bearbeitet werden müssen, nennen wir **aktive** Branch&Cut-Knoten und die schon bearbeitet wurden, **nicht aktive** Branch&Cut-Knoten.

Für die Teilprobleme werden mittels dualer Heuristiken **lokale untere Schranken** (englisch: local lower bound, kurz: llb) berechnet. Die kleinste lokale untere Schranke von allen aktiven Branch&Cut-Knoten und des aktuellen Branch&Cut-Knotens ist die **globale untere Schranke** für den Wert der Optimallösung des Originalproblems. Außerdem berechnen wir mittels primaler Heuristiken (Eröffnungs- und Verbesserungsheuristiken) **globale obere Schranken** (englisch: global upper bound, kurz: gub) für den Optimalwert des Originalproblems. Diese unteren und oberen Schranken (**bounding**) dienen zur Reduzierung der Anzahl der enumerativen Schritte.

Ein wesentlicher Teil eines erfolgreichen B&C-Algorithmus ist die Berechnung der unteren Schranken. Als duale Heuristik dient uns hier das Schnittebenenverfahren. Durch das Weglassen der Ganzzahligkeitsbedingungen in der ILP-Formulierung von $(E, \mathcal{I}, c)$ erhalten wir eine LP-Relaxierung entsprechend der Definition 3.1. Durch das Hinzufügen weiterer gültiger Ungleichungen verstärken wir diese Relaxierung und nähern uns somit an das Polytop $P_{\mathcal{I}}$ (2.1) an. Für die Teilprobleme werden weitere Schnittebenen generiert, um gebrochene Lösungen "abzuschneiden" (**cutting**). Erhalten wir eine zulässige Lösung, d.h., die Lösung ist ein Element von $\mathcal{I}$, so haben wir für das Teilproblem eine Optimallösung gefunden. Andernfalls ist der Lösungswert eine untere Schranke für den Optimalwert des Teilproblems.

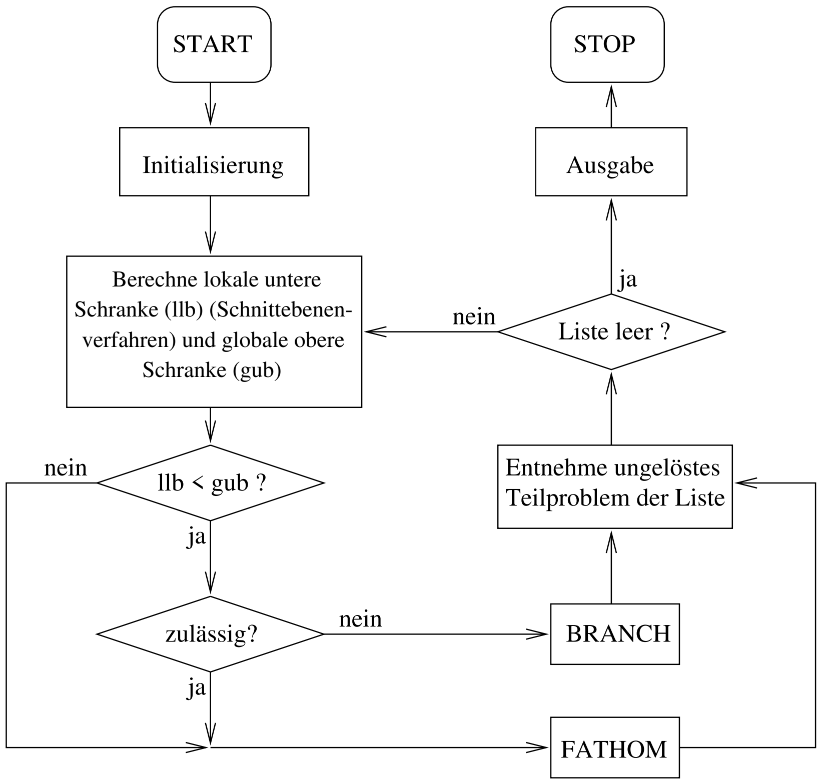

Abbildung 3.2 Flußdigramm eines Branch&Cut-Algorithmus

Die Abbildung 3.2 zeigt ein vereinfachtes Flußdiagramm eines B&C-Algorithmus, anhand dessen die Vorgehensweise des Algorithmus erklärt werden kann. Ein B&C-Algorithmus verwaltet eine Liste von ungelösten Teilproblemen, welche mit dem Originalproblem initialisiert wird. In jeder Iteration entnimmt der Algorithmus der Liste ein ungelöstes Teilproblem und wendet das Schnittebenenverfahren an, welches eine Lösung x^* produziert. Die folgenden Möglichkeiten können dabei auftreten:

- Die Lösung x^* ist zulässig bzgl. des Teilproblems. In diesem Fall haben wir eine Optimallösung für dieses Teilproblem gefunden. Damit ist dieses Teilproblem "ergründet" (englisch: **fathomed**). Ist der Lösungswert $c^T x^*$

kleiner als die globale obere Schranke, so wird die zu der globalen oberen Schranke gehörende zulässige Lösung durch die zulässige Lösung des Teilproblems ersetzt.

- Der Lösungswert $c^T x^*$ ist als lokale untere Schranke mindestens so groß wie die globale obere Schranke. Somit kann dieses Teilproblem keine bessere als die bereits vorhandene zulässige Lösung liefern und gilt als "fathomed".

- Der Lösungswert $c^T x^*$ ist als lokale untere Schranke kleiner als die globale obere Schranke, und x^* ist nicht zulässig bzgl. des Teilproblems. In diesem Fall wird das Teilproblem in zwei weitere Teilprobleme unterteilt, welche der Liste der ungelösten Teilprobleme hinzugefügt werden (branch). Außerdem wird noch versucht, aus der Lösung der LP-Relaxierung eine zulässige Lösung für das Originalproblem zu konstruieren, um die globale obere Schranke zu verbessern.

Es kann auch der Fall eintreten, daß das Teilproblem keine zulässige Lösung enthält (z.B. durch das Setzen der Branching-Variablen). Dann braucht dieses Teilproblem ebenfalls nicht weiter betrachtet zu werden. Der B&C-Algorithmus terminiert, wenn die Liste der ungelösten Teilprobleme leer ist. Man beachte, daß die Menge der zulässigen Lösungen als endlich angenommen wurde. Die Invariante des Algorithmus besteht darin, daß in jedem Branching-Schritt, in welchem aus dem gerade betrachteten Teilproblem zwei neue Teilprobleme erzeugt werden, die Vereinigung der Mengen der zulässigen Lösungen der beiden neuen Teilprobleme alle zulässigen Lösungen des vorangegangenen Teilproblems enthält.

Da im Laufe des B&C-Algorithmus alle generierten Teilprobleme bearbeitet werden müssen, braucht man geeignete Enumerations-Strategien. Entsprechend der Repräsentation des B&C-Algorithmus als Entscheidungsbaum bieten sich hier Durchlaufstrategien wie **Depth-First-Search** oder **Breadth-First-Search** an (siehe hierzu [Jun90]).

Ist ein Teilproblem "ergründet" (fathomed), so braucht der an diesem Branch-&Cut-Knoten hängende Teilbaum nicht mehr betrachtet zu werden. Auf diese Weise verringert sich die Anzahl der enumerativen Schritte erheblich, und wir haben den Schritt von der vollständigen Enumeration zur impliziten Enumeration vollzogen.

4 Das ATSP mit Zeitfenstern

Nachdem im vorangegangenen Kapitel die mathematischen Grundlagen für die vorliegende Arbeit gelegt wurden, möchte ich in diesem Kapitel das ATSP mit Zeitfenstern (englisch: ATSP with time windows, kurz ATSPTW) als kombinatorisches Optimierungsproblem vorstellen.

Der erste Abschnitt dieses Kapitels ist eine kurze Einführung in das Thema der Travelling Salesman Probleme. Es wird das Problem vorgestellt und ein Überblick über das große Thema der Routing und Scheduling Probleme, zu dem das ATSPTW gezählt werden kann, gegeben. Im zweiten Abschnitt möchte ich anhand eines Beispiels aus der Praxis zeigen, wie kombinatorische Optimierungsprobleme zur Lösung praktischer Problemstellungen ihre Anwendung finden. Das Beispiel stellt gleichzeitig eine Motivation für die vorliegende Arbeit dar. Ausgehend vom klassischen ATSP, welches im dritten Abschnitt kurz behandelt wird, werden im vierten Abschnitt drei verschiedene mathematische Modellierungen für das ATSPTW vorgestellt.

4.1 Einführung

Das wohl bekannteste Problem in der Kombinatorischen Optimierung ist das Travelling Salesman Problem (TSP). Das Problem besteht darin, eine kürzeste Rundreise durch n Städte bei gegebenen Entfernungen zwischen den Städten zu finden (vgl. z.B. [Grö80]). Nun sind die Anwendungen im Industrie- oder Dienstleistungssektor derart vielseitig, daß die Bezeichnungen "Städte" und "Entfernungen" nur noch symbolischen Charakter haben. So läßt sich z.B. die Bestimmung einer guten oder optimalen Bohrreihenfolge zur Verkürzung der Produktionszeit beim Bohren von Leiterplatten als ein TSP modellieren. Die Städte entsprechen dann den Bohrlöchern und die Entfernungen nicht mehr dem euklidischen Abstand, sondern der Maximum-Norm (man denke an einen Bohrer, der durch zwei Motoren in horizontaler und vertikaler Richtung positioniert werden kann). Ganz allgemein läßt sich das TSP in graphentheoretischer Terminologie beschreiben.

Travelling Salesman Problem. *Es sei ein vollständiger Graph $G = (V, E)$, $|V| = n$, mit Kantengewichten c_e für alle $e \in E$ gegeben. Gesucht ist ein Hamiltonkreis in $G = (V, E)$ mit minimalem Gewicht.*

Analog formuliert man das TSP für einen Digraphen $D = (V, A)$ mit Bogengewichten c_{ij} für alle $(i, j) \in A$. Gilt für die Gewichte $c_{ij} = c_{ji}$, $1 \leq i < j \leq n$, so sprechen wir von einem **symmetrischen** TSP, andernfalls von einem **asymmetrischen** TSP. Die Gewichte c_{ij} werden auch häufig als Kosten bezeichnet. Da wir in Zukunft ausschließlich auf Digraphen arbeiten werden, gehen wir, ohne dies jedesmal explizit zu erwähnen, davon aus, daß Wege, Kreise und Hamiltonkreise bzw. Touren gerichtet sind.

Die in der Praxis auftretenden Probleme erfordern oft anwendungsspezifische Restriktionen. Soll z.B ein Knoten $i \in V$ vor einem Knoten $j \in V$ besucht werden, so reden wir von einer **Reihenfolgebedingung**. Es kann auch erforderlich sein, daß ein Knoten $i \in V$ innerhalb eines bestimmten Zeitintervalls $[r_i, d_i]$ besucht werden muß. Ein solches Zeitintervall heißt **Zeitfenster**, und wir reden dann von einer **Zeitfensterrestriktion**. Diese Zeitfensterrestriktionen führen zu einem Travelling Salesman Problem mit Zeitfenstern (TSPTW) oder im asymmetrischen Fall zu einem ATSPTW.

Das ATSPTW läßt sich in die große Klasse der **Routing** und **Scheduling** Probleme einordnen. Derartige Probleme treten in vielfältiger Form im Güter- und Personenverkehr und im Rahmen der innerbetrieblichen Transport,- Lager- und Umschlagslogistik auf. Konkrete Beispiele sind hier die Routenplanung von Lieferwagen, von Straßenverkehrsmaschinen, der Müllabfuhr, von Speditionen und von Schulbussen und die Steuerung von Flexiblen Fertigungssystemen (FFS), wie z.B. die Steuerung von fahrerlosen Transportsystemen oder die Steuerung von Regalbediengeräten in Hochregallagern.

Unter den Routenplanungsproblemen mit Zeitfenstern gibt es zwei grundlegende Typen, das **Vehicle Routing Problem** mit Zeitfenstern (VRPTW) und das **Pickup-and-Delivery Problem** mit Zeitfenstern (PDPTW). Bei dem VRPTW haben wir eine Anzahl von Fahrzeugen in einem Depot gegeben und müssen eine Anzahl von Kunden beliefern. Jedes Fahrzeug hat eine bestimmte Kapazität, und jeder Kunde muß innerhalb eines gegebenen Zeitfensters beliefert werden. Das Ziel ist die Minimierung der Transportkosten. Der Spezialfall, bei dem alle m Fahrzeuge unbeschränkte Kapazität haben, wird **Multiple Travelling Salesman Problem** mit Zeitfenstern (m-TSPTW) genannt. Eine konkrete Anwendung des m-TSPTW ist das Schulbus-Routenplanungsproblem. Bei dem PDPTW haben wir ebenfalls ein Depot mit einer Anzahl von Fahrzeugen mit bestimmten Kapazitäten sowie eine Anzahl von verschiedenen Transportaufträgen gegeben. Ein Transportauftrag besteht darin, das zu transportierende Objekt (Verkehrsobjekt) an einem bestimmten Ort innerhalb eines gegebenen Zeitfensters abzuholen und innerhalb eines anderen gegebenen Zeitfensters an einen bestimmten Zielort zu befördern. Ziel ist auch hier die Minimierung der Transportkosten. Der Spezialfall, bei dem die Transportaufträge bzgl. der Verkehrsobjekte identisch sind, heißt **Dial-a-Ride Problem** mit Zeitfenstern. Eine typische Anwendung dieses Problems ist die Beförderung von älteren oder behinderten Personen. Ein umfassenderer Überblick über Routing Probleme wird in [AG88] gegeben.

Das Travelling Salesman Problem mit Zeitfenstern (TSPTW) ist ein VRPTW mit nur einem Fahrzeug ohne Kapazitätsbeschränkungen. Da schon die Frage der Existenz einer zulässigen Lösung für das TSPTW $\mathcal{NP}$-vollständig ist (siehe

[Sav85]), sind in der Literatur zur exakten Lösung des TSPTW vor allem implizite Enumerationsverfahren, wie die Dynamische Programmierung oder das Branch&Bound-Verfahren, zu finden. In [DDSS95] sind weitere Lösungsansätze für das TSPTW aufgeführt.

4.2 Ein Beispiel aus der Praxis

Die vorliegende Arbeit hat ihren Ursprung in einem Projekt mit der Siemens Nixdorf Informationssysteme AG (SNI) zur Steuerung eines **Flexiblen Fertigungssystems** (FFS), welches zur Produktion von Personalcomputern (PC) dient. Das FFS besteht im wesentlichen aus computergestützten Produktionslinien, einem vollautomatischen innerbetrieblichen Transportsystem sowie einem vollautomatischen Lagerungssystem, bestehend aus mehreren Hochregallagern. Ziel des Projektes war die Optimierung des Materialflusses durch das gesamte System vom Input der Produktionsfaktoren bis hin zum Output der Fertigprodukte zur Erhöhung des Produktionsvolumens.

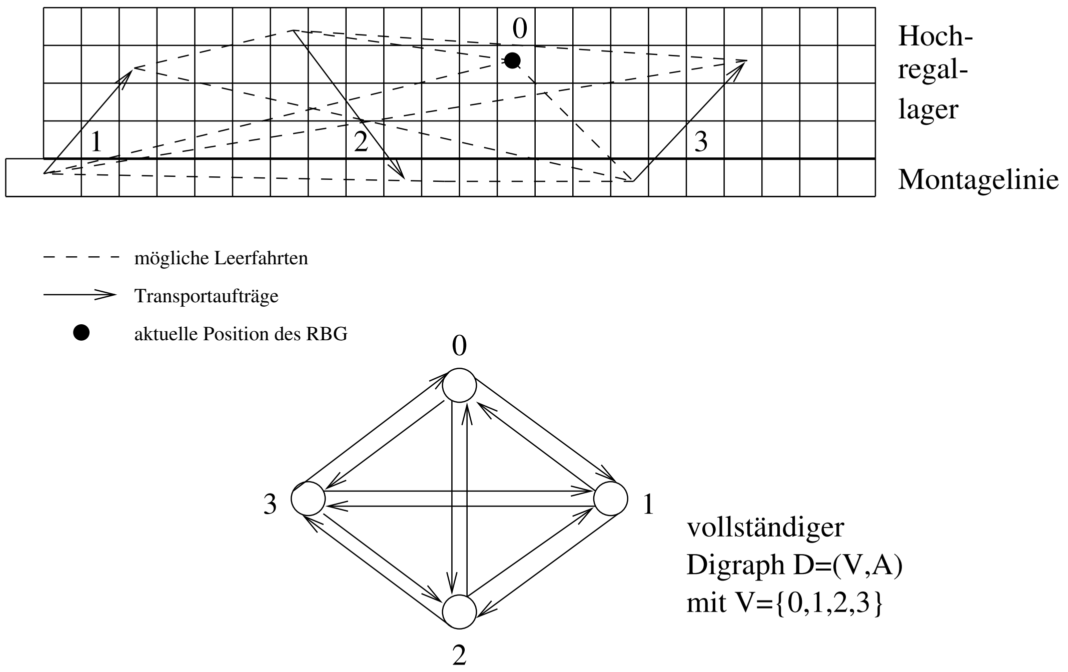

Abbildung 4.1 Minimierung der Leerfahrten eines RBG

Ein Teilproblem bei der Steuerung des FFS ist die Optimierung der Fahrten der Regalbediengeräte (RBG) an den Hochregallagern. Dabei sollen die Transportaufträge (Ein-, Aus- und Umlagerungen) für ein RBG in einer Reihenfolge ausgeführt werden, so daß die Zeiten der Leerfahrten des RBG minimiert werden. Dieses Problem kann mathematisch als ein asymmetrisches Hamilton'sches Wege-Problem (englisch: Asymmetric Hamiltonian Path Problem, kurz: AHPP) modelliert werden. Ganz allgemein läßt sich das AHPP graphentheoretisch wie folgt formulieren.

Asymmetrisches Hamilton'sches Wege-Problem. *Es sei ein vollständiger Digraph $D = (V, A)$, $|V| = n$, mit Bogengewichten c_{ij} für alle Bögen $(i, j) \in A$ gegeben. Gesucht ist ein Hamiltonweg in $D = (V, A)$ mit minimalem Gewicht.*

Das Problem der Minimierung der Leerfahrzeiten eines RBG bei der Abarbeitung von n Transportaufträgen läßt sich nun wie folgt als AHPP modellieren. Die Abbildung 4.1 stellt die Modellierung schematisch dar. Mit jedem Transportauftrag und mit der aktuellen Position des RBG assoziieren wir einen Knoten in einem vollständigen Digraphen $D = (V, A)$ mit $|V| = n+1$. Die aktuelle Position des RBG sei durch den Knoten 0 und die Transportaufträge $1, \ldots, n$ durch die Knoten $1, \ldots, n$ repräsentiert. Es sei $Z(i, j)$ die notwendige Leerfahrzeit, wenn Knoten j unmittelbar nach Knoten i besucht wird. Die Bogengewichte c_{ij} seien dann wie folgt erklärt:

$$c_{ij} = \begin{cases} Z(i,j) & \text{falls } i, j \in V, i \neq j, j \neq 0 \\ \infty & \text{falls } j = 0, i \in V \setminus \{0\}. \end{cases}$$

Damit ist garantiert, daß ein optimaler Hamiltonweg im Knoten 0 beginnt. Aus der Abbildung 4.1 ist ersichtlich, warum wir es mit einem *asymmetrischen* Hamilton'schen Wege-Problem zu tun haben. Betrachtet man beispielsweise die Transportaufträge 2 und 3, so gilt offensichtlich $c_{23} \neq c_{32}$.

Man beachte, daß das beschriebene Modell zur Optimierung der Fahrten eines RBG statischen Charakter hat. Dynamische Veränderungen des Systems während der Ausführung von Transportaufträgen werden nicht berücksichtigt. Die Praxis erfordert jedoch ein dynamisches Modell, welches Änderungen des laufenden Systems berücksichtigt. Damit wird das Problem zu einem **Online-Optimierungsproblem**, zu dessen Lösung wir einen Online-Algorithmus benötigen. Die Besonderheit eines Online-Algorithmus besteht darin, daß die Eingabedaten für den Algorithmus im vornherein nicht vollständig beschrieben werden können. In der Doktorarbeit von Ascheuer [Asc95] werden verschiedene Online-Heuristiken für das Online-Optimierungsproblem vorgestellt und miteinander verglichen. Das im Rahmen des Projektes bei der SNI entwickelte Optimierungspaket führt im Hochlastbetrieb zu einer Reduzierung der Leerfahrzeiten um bis zu 30%.

Online-Heuristiken liefern **obere Schranken** für den Wert der Optimallösung des Online-Optimierungsproblems. Um nun eine Gütegarantie für den Wert der Lösung einer Online-Heuristik anzugeben, benötigen wir zusätzlich als Bezugswerte **untere Schranken** für den Wert der Optimallösung des Online-Optimierungsproblems. Derartige untere Schranken erhält man durch eine sogenannte **Offline-Analyse**. Für die Optimierung der Fahrten eines RBG bedeutet das, daß wir über eine längere Zeitperiode hinweg alle eingehenden Transportaufträge sammeln und für diese durch das Lösen des AHPP eine optimale Reihenfolge der Ausführung ermitteln. Der Wert dieser Optimallösung stellt dann eine untere Schranke für den Wert der Optimallösung des Online-Optimierungsproblems dar. Allerdings erhalten wir mit diesem Modell eine sehr schlechte untere Schranke, da die zeitliche Komponente des Systems unberücksichtigt bleibt. So kann ein Transportauftrag j, der zeitlich nach Transportauftrag i generiert wurde, nicht vor diesem ausgeführt werden.

Um nun die zeitliche Komponente des Systems zu berücksichtigen und damit bessere untere Schranken zu erhalten, betrachten wir zusätzlich für jeden Transportauftrag i ein Zeitfenster $[r_i, d_i]$. Hierbei ist r_i der frühestmögliche Zeitpunkt (englisch: release date) und d_i der spätestmögliche Zeitpunkt (englisch: due date oder auch deadline) des Beginns der Ausführung des Transportauftrages i. Außerdem benötigt jeder Transportauftrag i eine Ausführungszeit $p_i \geq 0$. Damit erhalten wir ein **AHPP mit Zeitfenstern** (AHPPTW). Dieses Modell liefert deutlich bessere untere Schranken als das AHPP ohne Zeitfenster. Jedoch führen die Zeitfensterrestriktionen zu einem erhöhten Schwierigkeitsgrad, so daß nicht jedes Problembeispiel in vernünftiger Zeit optimal gelöst werden kann. Daher greift man auch auf eine Relaxierung des AHPPTW, das sogenannte Sequential Ordering Problem (SOP), zurück. Das SOP ist ein AHPP mit Reihenfolgebedingungen. Die Reihenfolgebedingungen werden von den Zeitfensterrestriktionen impliziert. So löst man anstelle des AHPPTW das SOP und erhält ebenfalls eine untere Schranke für den Wert der Optimallösung des Online-Optimierungsproblems.

Zur Lösung des AHPPTW wurde von Ascheuer [Asc95] ein Branch&Cut-Algorithmus implementiert, welcher zwei verschiedene mathematische Modellierungen des AHPPTW alternativ verwenden kann. Da das AHPP auf einfache Weise in ein ATSP und das ATSP auf einfache Weise in ein AHPP transformierbar ist, kann der Algorithmus nach entsprechenden Modifikationen auch zur Lösung des ATSPTW verwendet werden.

Die Transformation eines AHPP in ein ATSP läßt sich wie folgt beschreiben. Sei $C = (c_{ij})$ eine $n \times n$-Kostenmatrix und $D = (V, A)$ mit $|V| = n$ ein vollständiger Digraph. Angenommen, wir suchen in D einen kostenminimalen gerichteten Hamiltonweg, wobei ein Anfangsknoten s und ein Endknoten t fest vorgegeben sind. Dann ist das AHPP äquivalent zu einem ATSP auf $n-1$ Knoten, bei dem die Knoten s und t zu einem Knoten, sagen wir d, zusammengeschmolzen sind und die Kostenmatrix $C' = (c'_{ij})$ durch

$$c'_{ij} = \begin{cases} c_{sj} & \text{falls } i = d \\ c_{it} & \text{falls } j = d \\ c_{ij} & \text{sonst} \end{cases}$$

gegeben ist. Angenommen, wir suchen in D einen kostenminimalen gerichteten Hamiltonweg, ohne daß Anfangs- und Endknoten des Weges fest gegeben sind. Dann ist dieses Problem äquivalent zu einem ATSP auf $n+1$ Knoten, dessen Kostenmatrix wir aus C durch Hinzufügen einer $(n+1)$-ten Zeile und Spalte mit Kosten 0 erhalten.

Wollen wir nun umgekehrt ein ATSP in ein AHPP transformieren, so gehen wir wie folgt vor. Es sei ein ATSP auf n Knoten durch die $n \times n$-Kostenmatrix $C = (c_{ij})$ gegeben. Ein AHPP, bei dem ein Anfangsknoten s und ein Endknoten t fest vorgegeben sind, erhalten wir nun dadurch, indem wir von dem Knoten s eine Kopie t machen und einen kostenminimalen gerichteten Hamiltonweg von s nach t suchen. Darf hingegen kein Anfangs- bzw. Endknoten fest vorgegeben sein, so können wir dennoch durch das Setzen der Kosten $c_{is} = M$ und $c_{tj} = M$, wobei M eine genügend große Zahl ist, die Knoten s bzw. t dazu bringen, Anfangs- bzw. Endknoten eines kürzesten Hamiltonweges zu sein.

Die Schwierigkeiten bei der Lösung des AHPPTW, die sich unter anderem dadurch bemerkbar machten, daß nicht alle Problembeispiele in einer vernünftigen Zeit optimal gelöst werden konnten, führten zu dem Entschluß, dieses Problem im Rahmen von ATSPTW weitergehend zu untersuchen.

4.3 Das klassische ATSP

In diesem Abschnitt wollen wir für das klassische ATSP ein binäres lineares Programm aufstellen, da dieses die Grundlage für die mathematischen Modellierungen des ATSPTW bildet. Das ATSP können wir graphentheoretisch wie folgt formulieren:

- gegeben:
  - vollständiger Digraph $D = (V, A)$ mit $V := \{1, \dots n\}$ und $A := \{(i,j) \,|\, i, j \in V, i \neq j\}$,
  - Kosten $c_{ij} \geq 0 \; \forall \, (i,j) \in A$,
- gesucht:
  - Tour (gerichteter Hamiltonkreis) mit minimalen Kosten.

Sei nun $T \subseteq A$ eine Tour in $D = (V, A)$. Für jeden Bogen $(i, j) \in A$ führen wir dann eine binäre Variable x_{ij} mit

$$x_{ij} = \begin{cases} 1 & \text{falls } (i,j) \in T \\ 0 & \text{falls } (i,j) \notin T \end{cases}$$

ein. Für eine Bogenmenge $B \subseteq A$ sei $x(B) := \sum_{(i,j)\in B} x_{ij}$. Vor diesem Hintergrund sind wir nun in der Lage, das ATSP als binäres lineares Programm zu formulieren:

$$\begin{array}{llrcll} \textbf{ATSP}: & & \min c^T x & & & \\ & (1) & x(\delta^+(i)) & = & 1 & \forall i \in V \\ & (2) & x(\delta^-(i)) & = & 1 & \forall i \in V \\ & (3) & x(A(W)) & \leq & |W| - 1 & \forall\, W \subset V, \\ & & & & & 2 \leq |W| \leq n-1 \\ & (4) & x_{ij} & \in & \{0,1\} & \forall (i,j) \in A. \end{array} \tag{4.1}$$

Da wir später zur Lösung des ATSPTW im Rahmen eines Branch&Cut-Algorithmus das Schnittebenenverfahren verwenden wollen, interessieren wir uns hier besonders für die polyederische Struktur des Systems (4.1) (1)–(4). Durch

$$P_{ATSP} := \text{conv}\{x \in \mathbb{R}^A \mid x \text{ erfüllt (4.1) (1)–(4)}\}$$

definieren wir das ATSP–Polytop, dessen Ecken genau die Inzidenzvektoren von Touren in $D = (V, A)$ sind. Das Polytop P_{ATSP} ist in dem Schnitt der Hyperebenen, welche durch die Gleichungen in (1) und (2) definiert werden, enthalten. Diese Gleichungen heißen **Gradbeschränkungen** und modellieren die grundlegende Eigenschaft einer gerichteten Tour in einem Digraphen, die darin besteht, daß aus jedem Knoten $i \in V$ genau ein Bogen heraus (1) und genau ein Bogen herein (2) geht. Da auch gerichtete Kreise der Länge $k < n$ diese Eigenschaft besitzen, müssen wir derartige Kreise, auch **Subtouren** genannt, ausschließen. Dies geschieht durch die Ungleichungen (3), die sogenannten **Kurzzyklus-Ungleichungen** (englisch: Subtour Elimination Constraints, kurz: SEC). Die Bedingung (4) ist die **Ganzzahligkeitsbedingung**.

Aus polyedertheoretischer Sicht und mit Hinblick auf das Schnittebenenverfahren sind auch die Kenntnis der Dimension des zu untersuchenden Polytops sowie die Frage, ob Ungleichungen Facetten definieren oder nicht, von Bedeutung. Für ein Polyeder $P \subseteq \mathbb{R}^n$ bezeichnen wir im folgenden mit $\dim(P)$ dessen Dimension.

Satz 4.1
(i) $\dim(P_{ATSP}) = |A| - 2|V| + 1$.
(ii) Die SEC definieren Facetten bezüglich P_{ATSP} $(|V| \geq 5)$.

Beweis:
Siehe [GP85].

□

Das System von Gleichungen und Ungleichungen in (4.1) enthält redundante Bedingungen. So kann eine der Gleichungen in (1) und (2) weggelassen werden, ohne daß sich das Polytop P_{ATSP} ändert. Darüber hinaus sind zwei SEC $x(A(W_1)) \leq |W_1| - 1$ und $x(A(W_2)) \leq |W_2| - 1$ äquivalent, falls $W_2 = V \backslash W_1$.

Läßt man die SEC aus (4.1) weg, so erhält man eine Formulierung des **Zuordnungsproblems** als ganzzahliges lineares Programm. Das Zuordnungsproblem ist somit eine Relaxierung des ATSP. Das Problem besteht darin, in einem vollständigen Digraphen $D = (V, A)$ mit Bogengewichten $c_a \geq 0$ für alle $a \in A$ eine Bogenmenge $B \subseteq A$ minimalen Gewichts zu finden, so daß jeder Knoten jeweils Anfangs- und Endknoten genau eines Bogens aus B ist. Eine Zuordnung ist somit die Vereinigung knotendisjunkter gerichteter Kreise. Jede Tour ist also eine Zuordnung, aber nicht umgekehrt. Für das Zuordnungsproblem gibt es effiziente Algorithmen, die das Problem in polynomialer Zeit (z.B. $O(n^3)$) lösen.

Es sei noch bemerkt, daß die Ganzzahligkeitsbedingungen für das Zuordnungsproblem relaxiert werden können, da die Matrix der Nebenbedingungen (4.1) (1) und (2) **total unimodular** ist, d.h., jede quadratische Untermatrix hat Determinante 0, 1 oder -1. Nach einem Resultat von Hoffman und Kruskal (1956) sind somit die Ecken des Polytops $\{x \in \mathbb{R}^A \,|\, x \text{ erfüllt (4.1) (1) und (2)}, x \geq 0\}$ ganzzahlig.

4.4 Mathematische Modellierungen des ATSPTW

In diesem Abschnitt werden drei verschiedene mathematische Modellierungen für das ATSPTW vorgestellt. Wie schon erwähnt wurde, ist das ATSP ein $\mathcal{NP}$-schweres Problem. Die Zeitfensterrestriktionen beim ATSPTW führen darüber hinaus zu einer erhöhten Schwierigkeit bei der Lösung des Problems. Das Standardmodell für das ATSPTW (vgl. Modellierung 2), wie es häufig in der Literatur zu finden ist, hat auf Grund der Verwendung von "big M" aus nummerischer Sicht sehr schlechte Eigenschaften. Dies gab den Anstoß dazu, weitere Modellierungen des ATSPTW heranzuziehen und miteinander zu vergleichen.

Wie wir im Abschnitt 2.2 gesehen haben, lassen sich kombinatorische Optimierungsprobleme auf relativ einfache Weise als ganzzahlige lineare Programme darstellen. Die entsprechende LP-Relaxierung kann dann als Anfangs-LP für das Schnittebenenverfahren innerhalb eines B&C-Algorithmus verwendet werden. Somit führen verschiedene mathematische Modellierungen des ATSPTW in Form von ganzzahligen linearen Programmen zu einem unterschiedlichen Laufzeitverhalten eines B&C-Algorithmus.

Zunächst geben wir eine Formulierung des ATSPTW in graphentheoretischer Terminologie an. In den drei nachfolgenden Unterabschnitten werden dann die verschiedenen mathematischen Modellierungen des ATSPTW vorgestellt. Graphentheoretisch läßt sich das ATSPTW wie folgt formulieren:

- gegeben:
  - vollständiger Digraph $D = (V \cup \{0\}, A)$ mit $V := \{1, \dots n\}$ und $A := \{(i,j) \,|\, i, j \in V \cup \{0\}, i \neq j\}$,
  - Kosten $c_{ij} \geq 0 \; \forall \, (i,j) \in A$,
  - Ausführungszeiten $p_i \geq 0 \; \forall i \in V$,
  - Zeitfenster $[r_i, d_i]$ mit $r_i \geq 0$ und $d_i \geq r_i \; \forall i \in V$,
- gesucht:
  - kostenminimale Tour (gerichteter Hamiltonkreis), so daß der Beginn der Ausführung jedes Knotens $i \in V$ innerhalb seines Zeitfensters $[r_i, d_i]$ liegt.

Wir nehmen an, daß jede Tour im Knoten 0 beginnt und endet. Somit wird der Knoten 0 auch als **Depotknoten** bezeichnet. Der Zeitpunkt r_i ist der frühestmögliche (englisch: **release date**) und der Zeitpunkt d_i der spätestmögliche (englisch: **due date**) Beginn der Ausführung des Knotens $i \in V$. Wir nehmen im weiteren an, daß c_{ij} für alle $(i,j) \in A$ sowie p_i, r_i und d_i für alle $i \in V$ ganzzahlige Werte sind. Die Kosten c_{ij} können z.B. als Zeiteinheiten betrachtet werden, die benötigt werden, um vom Knoten i zum Knoten j zu gelangen. Mit

$$\vartheta_{ij} := p_i + c_{ij}$$

bezeichnen wir im folgenden den notwendigen Zeitabstand zwischen dem Beginn der Ausführung des Knotens i und dem Beginn der Ausführung des Knotens j, wenn Knoten j unmittelbar nach Knoten i ausgeführt werden soll. Wir nennen einen Bogen $(i,j) \in A$ bzgl. der Zeitfenster **zulässig**, falls $r_i + \vartheta_{ij} \leq d_j$ gilt. Für die Zeitabstände ϑ_{ij} gelte die Dreiecksungleichung, d.h.

$$\vartheta_{ij} \leq \vartheta_{ik} + \vartheta_{kj} \;\; \forall i, j, k \in V, \; i \neq j \neq k \neq i.$$

Für einen Knoten $i \in V$ heißt das Zeitfenster **aktiv**, falls $r_i > 0$ oder $d_i < \infty$. Die **Weite** eines Zeitfensters ist durch $d_i - r_i$ gegeben. Ein Zeitfenster $[0, \infty)$ heißt **relaxiert**. In der Literatur wird auch häufig zwischen **harten** und **weichen** Zeitfenstern unterschieden. In unserem Fall betrachten wir ausschließlich harte Zeitfenster, d.h., im Falle des Eintreffens an einem Knoten $i \in V$ vor dem

Zeitpunkt r_i muß mit der Ausführung des Knotens i bis zum Zeitpunkt r_i gewartet werden. Derartige Wartezeiten sind in unserem Fall zugelassen. Im Gegensatz zu harten Zeitfenstern ist das Verletzen von weichen Zeitfenstern bei Inkaufnahme von Strafkosten erlaubt.

In der obigen graphentheoretischen Formulierung des ATSPTW wird als Zielfunktion die Minimierung der Kosten betrachtet. In vielen Fällen, insbesondere im Rahmen von Scheduling Problemen (jeder Knoten entspricht z.B. einem Auftrag), ist aber auch die Minimierung der Fertigstellungszeit des letzten Knotens (**Makespan**), d.h. der Differenz zwischen dem Zeitpunkt der Fertigstellung des letzten Knotens und dem Zeitpunkt des Beginns der Ausführung des ersten Knotens, von Interesse. Eine andere Fragestellung wäre auch die Minimierung der Summe der Wartezeiten an den Knoten (z.B. Kunden), d.h. der Summe der Zeiten vom Öffnen eines Zeitfensters bis zum Beginn der Ausführung des jeweiligen Knotens.

Man beachte, daß das ATSP ein Spezialfall des ATSPTW ist, bei dem alle Zeitfenster relaxiert und die Ausführungszeiten gleich Null sind.

4.4.1 Modellierung 1

Kommen wir nun zur ersten Modellierung des ATSPTW. Sie wurde ursprünglich von Ascheuer [Asc95] zur Modellierung des AHPPTW (siehe Abschnitt 4.2) entworfen und ist auf einfache Weise auf das ATSPTW übertragbar. Grundlage bildet die mathematische Modellierung des klassischen ATSP entsprechend Abschnitt 4.3. Die Zeitfensterrestriktionen werden durch eine zusätzliche Klasse von Ungleichungen, die sogenannten **Unzulässige-Wege-Beschränkungen** (englisch: infeasible path constraints), modelliert. Dazu benötigen wir einige Begriffe.

Definition 4.1
Wir sagen, ein Weg $P = (u_1, \dots, u_m)$ ist in einem Kreis $C = (v_1, \dots, v_k, v_1)$ mit $k \geq m$ enthalten, falls eine natürliche Zahl $l \in [0, k-m]$ existiert, so daß $v_{i+l} = u_i$ für alle $i = 1, \dots, m$ gilt.

Definition 4.2
Gegeben sei ein vollständiger Digraph $D = (V \cup \{0\}, A)$ mit Zeitfenstern $[r_i, d_i]$ für alle $i \in V$.
(i) Ein Hamiltonkreis, der im Depotknoten 0 beginnt und endet, heißt **zulässig**, *wenn jeder Knoten $i \in V$ innerhalb seines Zeitfensters besucht wird.*
(ii) Ein Weg $P = (v_1, \dots, v_k)$, $v_i \in V$, $2 \leq k \leq n$, heißt **unzulässig**, *wenn er in keinem zulässigen Hamiltonkreis enthalten ist.*

Um Unzulässige-Wege-Beschränkungen zu formulieren, benötigt man unzulässige Wege. Um festzustellen, ob ein Weg unzulässig ist, genügt oft das folgende

Lemma.

Lemma 4.1
Gegeben sei ein vollständiger Digraph $D = (V \cup \{0\}, A)$ mit Kosten c_{ij} für alle $(i,j) \in A$ sowie Ausführungszeiten p_i und Zeitfenster $[r_i, d_i]$ für alle $i \in V$. Für ϑ_{ij}, $(i,j) \in A$, gelte die Dreiecksungleichung. Dann ist ein Weg $P = (v_1, \ldots, v_k)$ mit $v_i \in V$ und $2 \leq k \leq n$ unzulässig, wenn eine der folgenden Bedingungen erfüllt ist:

(i) P verletzt das Zeitfenster für den Knoten v_k.

(ii) Es existiert ein Knoten $v_i \in V$, der nicht auf dem Weg P liegt, so daß die beiden Wege $P_1 = (v_i, v_1, \ldots, v_k)$ und $P_2 = (v_1, \ldots, v_k, v_i)$ die gegebenen Zeitfenster verletzen.

Beweis:
(i) Offensichtlich kann $P = (v_1, \ldots, v_k)$ in keinem zulässigen Hamiltonkreis enthalten sein, da das Zeitfenster für den Knoten v_k verletzt ist.
(ii) Sei $v_i \in V \backslash \{v_1, \ldots, v_k\}$ ein Knoten, so daß P_1 und P_2 die gegebenen Zeitfenster verletzten. Dann kann der Knoten v_i auf Grund der Dreiecksungleichung weder vor Knoten v_1 noch nach Knoten v_k innerhalb eines zulässigen Hamiltonkreises besucht werden. Somit kann P in keinem zulässigen Hamiltonkreis enthalten sein.

□

Mit Hilfe der eingeführten Unzulässige-Wege-Beschränkungen können wir nun für das ATSPTW ein binäres lineares Programm aufstellen. Für jeden Bogen $(i,j) \in A$ sei die binäre Variable $x_{ij} = 1$, falls der Bogen (i,j) in einer Tour enthalten ist, andernfalls sei $x_{ij} = 0$. Das ATSPTW läßt sich dann wie folgt als BLP formulieren:

$$\begin{array}{rrcll}
 & \min c^T x & & & \\
(1) & x(\delta^+(i)) & = & 1 & \forall i \in V \cup \{0\} \\
(2) & x(\delta^-(i)) & = & 1 & \forall i \in V \cup \{0\} \\
(3) & x(A(W)) & \leq & |W| - 1 & \forall W \subset V \cup \{0\}, \\
 & & & & 2 \leq |W| \leq n \\
(4) & x(P) & \leq & k-2 & \forall \text{ unzulässigen Wege} \\
 & & & & P = (v_1, \ldots, v_k),\ v_i \in V \\
(5) & x_{ij} & \in & \{0,1\} & \forall (i,j) \in A.
\end{array} \tag{4.2}$$

Die Gleichungen (1) und (2) entsprechen den Gradbeschränkungen und die Ungleichungen (3) den SEC, wie wir sie schon vom klassischen ATSP her kennen. Die Ungleichungen (4) entsprechen den Unzulässige-Wege-Beschränkungen. Durch sie schließen wir alle unzulässigen Wege aus, so daß der resultierende Hamiltonkreis keines der gegebenen Zeitfenster verletzt. Damit ist jede Lösung von (4.2) ein Inzidenzvektor eines zulässigen Hamiltonkreises und umgekehrt. Mit

$$P_1^{TW} := \text{conv}\{x \in \mathbb{R}^A \,|\, x \text{ erfüllt } (4.2)(1) - (5)\}$$

bezeichnen wir die konvexe Hülle der zulässigen Lösungen von (4.2). P_1^{TW} ist also das ATSPTW-Polytop, das auf der Modellierung 1 beruht. Man beachte, daß diese Modellierung nur zur Minimierung der Kosten, nicht aber zur Minimierung des Makespan verwendet werden kann, da hier keine Zeitvariablen vorhanden sind.

4.4.2 Modellierung 2

Die zweite Modellierung des ATSPTW ist ein Standardmodell, wie es häufig in der Literatur zu finden ist. Sie beruht auf einer von Miller, Tucker und Zemlin [MTZ60] stammenden alternativen Formulierung der SEC. Auch hier dient die Modellierung des klassischen ATSP als Grundlage. Jedoch werden die SEC, eine Klasse von Ungleichungen mit exponentieller Größenordnung, durch die sogenannten **Miller-Tucker-Zemlin-Kurzzyklus-Ungleichungen** (kurz: MTZ-Ungleichungen) ersetzt. Damit reduzierten Miller, Tucker und Zemlin die Kurzzyklus-Ungleichungen auf $|A|$ viele Ungleichungen. Allerdings benötigen wir dafür neben den binären Variablen x_{ij} für alle $(i,j) \in A$ zusätzliche Variablen t_i für alle $i \in V$. Die MTZ-Ungleichungen für das ATSP sind von folgender Gestalt:

$$t_i - t_j + (n-1)x_{ij} \leq n-2, \;\; i,j = 1,\dots,n, \; i \neq j, \tag{4.3}$$

$$1 \leq t_i \leq n-1, \;\; i = 1,\dots,n. \tag{4.4}$$

Im Gegensatz zu den SEC definieren diese Ungleichungen jedoch keine Facetten des Polytops, welches durch die konvexe Hülle der zulässigen Lösungen von (4.1) (1), (2) und (4) sowie (4.3) und (4.4) definiert wird. Somit erhält man eine schwächere LP-Relaxierung für das ATSP aus dieser Modellierung. Der Vorteil liegt jedoch darin, daß sich die Zeitfensterrestriktionen auf einfache Weise in die MTZ-Ungleichungen einbauen lassen.

Für das ATSPTW interpretieren wir die Variablen t_i als die Startzeit für die Ausführung des Knotens $i \in V$. Wir nehmen an, daß jede Tour zum Zeitpunkt 0 startet, d.h. $t_0 = 0$. Dann schreiben sich die MTZ-Ungleichungen für das ATSPTW als

$$x_{ij} > 0 \Rightarrow t_i + \vartheta_{ij} \leq t_j, \;\; i,j = 0,\dots,n, \; i \neq j, \; j \neq 0, \tag{4.5}$$

$$r_i \leq t_i \leq d_i, \;\; i = 1,\dots,n. \tag{4.6}$$

Um die Methoden der Linearen Programmierung anwenden zu können, brauchen wir die Bedingung (4.5) in linearer Form, d.h.

$$t_i + \vartheta_{ij} - (1 - x_{ij})M \leq t_j, \quad i, j = 0, \ldots, n, \, i \neq j, \, j \neq 0, \tag{4.7}$$

wobei M ein genügend großer Wert ("big M") ist. Für $x_{ij} = 1$ kommt die Ungleichung zum tragen, andernfalls ist sie auf Grund von M in trivialer Weise erfüllt. Nun können wir das ATSPTW wie folgt als ILP formulieren:

$$\begin{array}{lrcll}
 & \min c^T x & & & \\
(1) & x(\delta^+(i)) & = & 1 & \forall i \in V \cup \{0\} \\
(2) & x(\delta^-(i)) & = & 1 & \forall i \in V \cup \{0\} \\
(3) & t_i + \vartheta_{ij} - (1 - x_{ij})M & \leq & t_j & \forall (i,j) \in A, \; j \neq 0 \\
(4) & r_i \; \leq \; t_i & \leq & d_i & \forall i \in V \\
(5) & t_i & \in & \mathbb{N} & \forall i \in V \cup \{0\} \\
(6) & x_{ij} & \in & \{0,1\} & \forall (i,j) \in A.
\end{array} \tag{4.8}$$

Die Gleichungen (1) und (2) sind wieder die Gradbeschränkungen. Die Ungleichungen (3) entsprechen den MTZ-Ungleichungen. Sie verhindern das Auftreten von Subtouren. Wird der Knoten j unmittelbar nach Knoten i ausgeführt, d.h. $x_{ij} = 1$, so kommt die MTZ-Ungleichung für den Bogen (i, j) zum tragen. In diesem Fall muß die Startzeit t_j größer oder gleich der Startzeit t_i zuzüglich der Ausführungszeit p_i und der Kosten c_{ij} sein. Für $x_{ij} = 0$ ist die MTZ-Ungleichung für den Bogen (i, j) auf Grund von "big M" erfüllt. Die Bedingungen (4) entsprechen den Zeitfensterrestriktionen und die Bedingungen (5) und (6) den Ganzzahligkeitsbedingungen. Mit

$$P_2^{TW} := \text{conv}\{(x, t) \in \mathbb{R}^{A \times V \cup \{0\}} \,|\, (x, t) \text{ erfüllt } (4.8)(1) - (6)\}$$

bezeichnen wir das ATSPTW-Polytop, das auf der Modellierung 2 basiert.

Ein entscheidender Nachteil dieser Formulierung des ATSPTW liegt in der "big M"-Modellierung. Erfahrungsgemäß führt dies zu numerischen Problemen bei der Berechnung der Optimallösung der LP-Relaxierung. Um dem ein wenig entgegenzuwirken, kann man für jeden Bogen $(i, j) \in A$ ein individuelles "big M_{ij}" wählen, so daß $M_{ij} \geq d_i + \vartheta_{ij} - r_j$ gilt. Ein weiterer Nachteil ist die schwache Verknüpfung der x-Variablen mit den t-Variablen durch die MTZ-Ungleichungen. Die Modellierung 3 wird zeigen, daß diese Nachteile auf Kosten einer größeren Anzahl von Variablen beseitigt werden können.

4.4.3 Modellierung 3

Wir stellen nun eine Formulierung des ATSPTW vor, die auf eine Formulierung des Delivery Man Problem mit Zeitfenstern (DMPTW) als MIP von van Eijl [vE95] zurückgeht. Das DMPTW ist eine Variante des TSPTW, bei dem eine in einem gegebenen Depot startende Tour gesucht wird, welche die Summe der Wartezeiten der Kunden minimiert.

Die Modellierung verwendet zwei verschiedene Typen von Variablen. Zu jedem Bogen $(i,j) \in A$ gehört zum einen eine binäre Variable x_{ij} und zum anderen eine Zeitvariable t_{ij}. Die binäre Variable x_{ij} gibt wieder an, ob der Bogen (i,j) zu einer Tour gehört oder nicht. Für die Zeitvariable gilt

$$t_{ij} = \begin{cases} \text{Startzeit der Ausführung des Knotens } i & \text{falls } x_{ij} = 1 \\ 0 & \text{sonst.} \end{cases}$$

Wir nehmen an, daß jede Tour zum Zeitpunkt 0 startet, d.h. $t_{0j} = 0$ für alle $j \in V$. Die Variablen t_{0j} brauchen somit in der Modellierung nicht berücksichtigt zu werden. Wir formulieren nun das ATSPTW als MIP wie folgt:

$$\min c^T x$$

$$\begin{array}{llcll}
(1) & \sum\limits_{\substack{j=0\\ j\neq i}}^{n} x_{ij} & = & 1 & i = 0,\dots,n \\[2ex]
(2) & \sum\limits_{\substack{i=0\\ i\neq j}}^{n} x_{ij} & = & 1 & j = 0,\dots,n \\[2ex]
(3) & \sum\limits_{\substack{i=1\\ i\neq j}}^{n} t_{ij} + \sum\limits_{\substack{i=0\\ i\neq j}}^{n} \vartheta_{ij} x_{ij} & \leq & \sum\limits_{\substack{k=0\\ k\neq j}}^{n} t_{jk} & j = 1,\dots,n \\[2ex]
(4) & r_i x_{ij} \leq t_{ij} & \leq & d_i x_{ij} & i,j = 0,\dots,n,\ i \neq j,\ i \neq 0 \\[2ex]
(5) & x_{ij} & \in & \{0,1\} & i,j = 0,\dots,n,\ i \neq j.
\end{array} \qquad (4.9)$$

Die Gleichungen (1) und (2) sorgen dafür, daß jeder Knoten einschließlich des Depotknotens genau einmal besucht wird (Gradbeschränkungen). Die Ungleichungen (3) korrespondieren mit den MTZ-Ungleichungen aus der Modellierung 2. Jedoch benötigen wir hier auf Grund der Verknüpfung der x- und t-Variablen kein "big M" zur Darstellung der MTZ-Ungleichungen in linearer Form. Ist $x_{ij} = 1$, so muß die Startzeit der Ausführung des Knotens j größer oder gleich der Startzeit der Ausführung des Knotens i zuzüglich ϑ_{ij} sein. Damit werden durch die Ungleichungen (3) Subtouren vermieden. Die Bedingungen (4) garantieren, daß $t_{ij} = 0$, falls $x_{ij} = 0$, und daß die Startzeit der Ausführung des Knotens i innerhalb des Zeitfensters $[r_i, d_i]$ liegt. Die Bedingungen (5) sind die Ganzzahligkeitsbedingungen. Mit

$$P_3^{TW} := \text{conv}\{(x,t) \in \mathbb{R}^{A \times A} \,|\, (x,t) \text{ erfüllt } (4.9)(1) - (5)\}$$

bezeichnen wir das zu der Modellierung 3 gehörige ATSPTW-Polytop.

Ein entscheidender Vorteil dieser Modellierung liegt darin, daß wir auf Grund der Verknüpfung der x- und t-Variablen kein "big M" mehr benötigen. Hatten wir jedoch bei der Modellierung 2 nur $|V|$ viele Zeitvariablen, benötigen wir hier $|A|$ viele Zeitvariablen, was nicht unerhebliche Auswirkung auf die Rechenzeit bei der Lösung der LP-Relaxierung haben dürfte. Numerische Probleme könnten darüber hinaus bei den Bedingungen (4) auftreten, falls einige der Zeitfenster relaxiert sind.

5 Schnittebenen für das ATSPTW

Wie wir im Kapitel 3 gesehen haben, dient uns das Schnittebenenverfahren in einem B&C-Algorithmus zur Bestimmung der unteren Schranken. Dazu ist die Kenntnis von problemspezifischen Schnittebenen von großer Bedeutung. Dieses Kapitel gibt einen Überblick über gültige Ungleichungen bzgl. der Polytope P_1^{TW}, P_2^{TW} und P_3^{TW}, die als Schnittebenen verwendet werden können. Auf Schnittebenen, die in dem B&C-Algorithmus verwendet werden, werde ich ausführlicher eingehen.

Große Klassen gültiger Ungleichungen für das symmetrische TSP können wir auch für das ATSP verwenden. Jede gültige Ungleichung für das ATSP ist ebenso eine gültige Ungleichung für das ATSPTW, da jede zulässige Lösung des ATSPTW, d.h. jeder zulässige Hamiltonkreis, auch eine Lösung des ATSP ist. Somit sind gültige Ungleichungen für das ATSP auch gültige Ungleichungen bzgl. unserer Polytope P_1^{TW}, P_2^{TW} und P_3^{TW}. Gültige Ungleichungen für das ATSP werden in den Abschnitten zwei und drei aufgeführt. Zuvor möchte ich im ersten Abschnitt die Methode der sequentiellen Liftung vorstellen. Mit dieser Methode können wir bekannte Facetten eines Polytops mit niedrigerer Dimension zu Facetten des eigentlich interessierenden Polytops erweitern. Sie ist insbesondere für den asymmetrischen Fall von Bedeutung, da hier neue facettendefinierende Ungleichungen erzeugt werden können. In den Abschnitten vier bis sechs werde ich Ungleichungen vorstellen, die insbesondere die Nebenbedingungen der Zeitfenster berücksichtigen. Dazu zählen die Vorgänger-, Nachfolger- bzw. die Vorgänger-Nachfolger-Ungleichungen (π-, σ- bzw. (π, σ)-Ungleichungen), die Unzulässige-Wege-Beschränkungen und Ungleichungen, die explizit Zeitvariablen enthalten, d.h. verschärfte Versionen der MTZ-Ungleichungen, geliftete Zeitfensterrestriktionen (englisch: lifted t-bounds) und sogenannte 2-Job-Ungleichungen (englisch: two-job-cuts).

5.1 Sequentielle Liftung

In diesem Abschnitt wollen wir die Methode der sequentiellen Liftung vorstellen. Anhand der sogenannten Kreis-Ungleichungen werden wir eine Anwendung dieser Methode beschreiben.

Es sei $D = (V, A), |V| = n$, ein vollständiger Digraph, $\mathcal{T}_n$ die Menge aller gerichteten Touren in D und $\tilde{\mathcal{T}}_n$ die Menge aller Teilmengen von Touren in $\mathcal{T}_n$. Mit

$$P_{ATSP} := \text{conv}\{\chi^T \in \mathbb{R}^A \,|\, T \in \mathcal{T}_n\}$$

bzw.

$$\tilde{P}_{ATSP} := \text{conv}\{\chi^S \in \mathbb{R}^A \,|\, S \in \tilde{\mathcal{T}}_n\}$$

bezeichnen wir das ATSP-Polytop bzw. das monotone ATSP-Polytop. Die Methode der sequentiellen Liftung führt im Gegensatz zum symmetrischen TSP beim ATSP zu neuen facettendefinierenden Ungleichungen für P_{ATSP} und $\tilde{P}_{ATSP}$. Zunächst wollen wir jedoch definieren, was wir unter einem Unabhängigkeitssystem verstehen.

Definition 5.1
Sei E eine endliche Menge und 2^E die Menge aller Teilmengen von E. Eine Menge $\mathcal{I} \subseteq 2^E$ heißt **Unabhängigkeitssystem** *auf E, falls $\mathcal{I}$ die beiden folgenden Bedingungen erfüllt:*
(i) $\emptyset \in \mathcal{I}$,
(ii) $F \subseteq G \in \mathcal{I} \Rightarrow F \in \mathcal{I}$.

Offensichtlich ist im Gegensatz zu $\mathcal{T}_n$ die Menge $\tilde{\mathcal{T}}_n$ ein Unabhängigkeitssystem. Sei E eine endliche Menge, $\mathcal{I} \subseteq 2^E$ und $P_{\mathcal{I}} := \text{conv}\{\chi^I \in \mathbb{K}^E \mid I \in \mathcal{I}\}$ (vgl. (2.1)). Für jede Teilmenge $F \subseteq E$ sei $P_{\mathcal{I}}(F) := \{x \in P_{\mathcal{I}} \mid x_e = 0 \; \forall e \in F\}$.

Satz 5.1
Seien $\mathcal{I}$ ein Unabhängigkeitssystem auf der endlichen Menge E, $F \subseteq E$ und $e \in F$. Angenommen, die Ungleichung $\sum_{k \notin F} a_k x_k \leq a_0$ definiert eine Facette von $P_{\mathcal{I}}(F)$ mit $a_0 > 0$, und es sei

$$a_e := a_0 - \max\{\sum_{k \notin F} a_k \chi_k^I \mid I \subseteq E \backslash F, \{e\} \cup I \in \mathcal{I}\}$$

der **Liftungskoeffizient** *der Variable x_e. Dann definiert $a_e x_e + \sum_{k \notin F} a_k x_k \leq a_0$ eine Facette von $P_{\mathcal{I}}(F \backslash \{e\})$.*

Beweis:
Siehe [Pad75].

□

Eine Tour in einem Digraphen $D = (V, A), |V| = n \geq 3$, ist offensichtlich ein Kreis der Länge n. Auf Grund dieser Eigenschaften können wir die **Kreis-Ungleichungen** formulieren. Sei $C \subseteq A$ die Bogenmenge eines gerichteten Kreises der Länge kleiner als n. Dann ist

$$x(C) \leq |C| - 1$$

eine gültige Ungleichung bzgl. $P_{ATSP}(F)$ und $\tilde{P}_{ATSP}(F)$. Darüber hinaus wurde in [GP85] folgendes Lemma bewiesen.

Lemma 5.1
Sei C die Bogenmenge eines gerichteten Kreises in $D = (V, A), |V| = n \geq 3$, der Länge $k \leq n - 1$, und sei $F := A \backslash C$. Dann definiert die Kreis-Ungleichung $x(C) \leq |C| - 1$ eine Facette von $\tilde{P}_{ATSP}(F)$. □

Da $\tilde{\mathcal{T}}_n$ als die Menge aller Teilmengen von Touren ein Unabhängigkeitssystem ist, kann nach Satz 5.1 und Lemma 5.1 jede Kreis-Ungleichung, die eine Facette von $\tilde{P}_{ATSP}(F)$ definiert, mittels sequentieller Liftung zu einer Facette von $\tilde{P}_{ATSP}$ erweitert ("geliftet") werden. Im analogen Fall des symmetrischen TSP erhalten wir nichts anderes als die uns bekannten Kurzzyklus-Ungleichungen. Für das ATSP erhalten wir neue Klassen von Ungleichungen.

Durch die sequentielle Liftung von Kreisen der Länge k (**k-Kreise**) erhalten wir beispielsweise die D_k^+- und die D_k^--Ungleichungen (siehe Abschnitt 5.3). In [Grö77] wurde bewiesen, daß diese Ungleichungen in den Fällen $k = 3$ und $k = 4$ Facetten für P_{ATSP} definieren. Wir betrachten hier den Fall $k = 3$. Die Abbildung 5.1 stellt eine gültige Ungleichung (geliftete 3-Kreis-Ungleichung) für das ATSP graphisch dar, die man durch sequentielle Liftung einer 3-Kreis-Ungleichung erhält. Dabei assoziieren wir mit den dünn gedruckten Bögen Variablen mit Koeffizient 1 und mit dem fett gedruckten Bogen eine Variable mit Liftungskoeffizient 2. Derartige geliftete Kreis-Ungleichungen werden wir für das Schnittebenenverfahren in unserem B&C-Algorithmus verwenden.

Lemma 5.2
Sei $D = (V \cup \{0\}, A)$ ein vollständiger Digraph, und seien $i, j, k \in V \cup \{0\}$. Dann ist die **geliftete 3-Kreis-Ungleichung**

$$x_{ij} + x_{jk} + x_{ki} + 2x_{ji} \leq 2$$

gültig bzgl. P_1^{TW}, P_2^{TW} und P_3^{TW}. □

Verletzte Ungleichungen dieser Art können durch Enumeration aller möglichen Knoten $i \neq j \neq k$ in $O(n^3)$ gefunden werden.

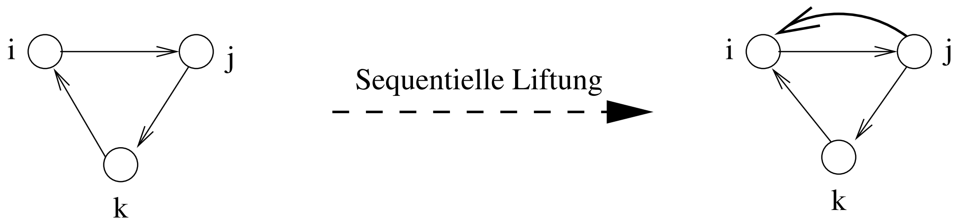

Abbildung 5.1 Sequentielle Liftung

In [GP85] sind weitere Beispiele für geliftete Kreis-Ungleichungen (z.B. für den Fall $k = 4$) zu finden. Schon diese wenigen Beispiele zeigen, wie kompliziert die sequentielle Liftung werden kann, was ein Hinweis auf die komplexe Struktur des ATSP-Polytops ist.

5.2 Symmetrische Ungleichungen

Das symmetrische TSP (STSP) ist ein Spezialfall des ATSP. Von besonderem Interesse sind daher die Beziehungen zwischen den gültigen Ungleichungen beider Probleme. Sei $D = (V, A), |V| = n$, ein vollständiger Digraph. Wir nennen eine ATSP-Ungleichung $a^T x \leq a_0$ **symmetrisch**, falls $a_{ij} = a_{ji}$ für alle $(i, j) \in A$, anderfalls nennen wir sie **asymmetrisch**. Symmetrische ATSP-Ungleichungen können von gültigen STSP-Ungleichungen abgeleitet werden. Sei $G = (V, E)$ ein vollständiger Graph und sei $y_e = 1$, falls $e \in E$ zu einer Tour in G gehört, andernfalls sei $y_e = 0$. Jede gültige STSP-Ungleichung $\sum_{e \in E} a_e y_e \leq a_0$ kann dann in eine gültige ATSP-Ungleichung transformiert werden, indem man $y_e = x_{ij} + x_{ji}$ für alle $e = ij \in E$ setzt. Damit erhalten wir eine symmetrische ATSP-Ungleichung $a^T x \leq a_0$, wobei $a_{ij} = a_{ji} = a_e$ für alle $e = ij \in E$. Umgekehrt korrespondiert jede symmetrische ATSP-Ungleichung $a^T x \leq a_0$ mit einer gültigen STSP-Ungleichung $\sum_{ij \in E} a_{ij} y_{ij} \leq a_0$. Beispiele für symmetrische ATSP-Ungleichungen sind

- Kurzzyklus-Ungleichungen,
- 2-Matching-Ungleichungen,
- Kamm-Ungleichungen,
- Cliquenbaum-Ungleichungen.

Einen Überblick über diese Klassen von Ungleichungen findet man in [GP85]. In unserem B&C-Algorithmus verwenden wir lediglich die Kurzzyklus-Ungleichungen und die 2-Matching-Ungleichungen.

Betrachten wir das Polytop, das durch die Gradbeschränkungen (4.1) (1) und (2) sowie durch die Ungleichungen $x_{ij} \geq 0$ für alle $(i, j) \in A$ gegeben ist, so sind dessen Ecken entweder Inzidenzvektoren von gerichteten Touren oder Inzidenzvektoren von disjunkten gerichteten Kreisen. Wie wir bereits wissen, können die Inzidenzvektoren von disjunkten gerichteten Kreisen (Subtouren) durch die Kurzzyklus-Ungleichungen abgeschnitten werden.

Lemma 5.3
Sei $D = (V \cup \{0\}, A), |V| = n$, ein vollständiger Digraph. Dann sind die **Kurzzyklus-Ungleichungen**

$$x(A(W)) \leq |W| - 1 \ \ \forall W \subset V, \, 2 \leq |W| \leq n$$

gültig bzgl. P_1^{TW}, P_2^{TW} und P_3^{TW}. □

Es ist offensichtlich, daß jeder Inzidenzvektor eines gerichteten Kreises eine der Kurzzyklus-Ungleichungen verletzt. Eine weitere Klasse von symmetrischen ATSP-Ungleichungen sind die 2-Matching-Ungleichungen.

Lemma 5.4
Sei $D = (V \cup \{0\}, A), |V| = n \geq 4$, ein vollständiger Digraph. Es seien außerdem $H, T_1, \ldots, T_k$ Teilmengen der Knotenmenge $V \cup \{0\}$ mit $k \geq 3$ ungerade oder $k = 1$ und $|H| \geq 4$, $|H \cap T_i| = 1$ und $|T_i \backslash H| = 1$ für $i = 1, \ldots, k$ sowie $T_i \cap T_j = \emptyset$ für $1 \leq i < j \leq k$. Dann heißt

$$x(A(H)) + \sum_{i=1}^{k} x(A(T_i)) \leq |H| + \frac{k-1}{2}$$

2-Matching-Ungleichung *und ist gültig bzgl. P_1^{TW}, P_2^{TW} und P_3^{TW}.* □

Der Schnitt der Halbräume, die durch die Kurzzyklus-Ungleichungen und die 2-Matching-Ungleichungen definiert werden, führt allerdings zu neuen gebrochenen Ecken. Diese können durch die Kamm-Ungleichungen und Cliquenbaum-Ungleichungen abgeschnitten werden.

Zur Lösung des Separierungsproblems für die Kurzzyklus- und die 2-Matching-Ungleichungen können wir Separierungsalgorithmen verwenden, die für das STSP entwickelt worden sind. Für einen gegebenen Punkt $x^* \in \mathbb{R}^A$ setzen wir

$$y_e^* := x_{ij}^* + x_{ji}^* \;\; \forall e = ij \in E$$

und wenden einen Separierungsalgorithmus für das STSP auf y^* an. Werden verletzte Ungleichungen gefunden, so transformieren wir diese in die entsprechenden symmetrischen ATSP-Ungleichungen. Zur Beschreibung exakter Separierungsalgorithmen für die Kurzzyklus- und die 2-Matching-Ungleichungen verweisen wir auf [PR90].

5.3 Asymmetrische Ungleichungen

Wie wir bereits im Abschnitt 5.1 gesehen haben, können wir durch sequentielle Liftung der Kreis-Ungleichung

$$x(C) \leq |C| - 1$$

neue facettendefinierende Ungleichungen für das ATSP-Polytop erhalten. Als Beispiel hatten wir bereits die **$\mathbf{D_k^+}$- und $\mathbf{D_k^-}$-Ungleichungen** erwähnt. Weitere Beispiele für Klassen von Ungleichungen, die wir durch sequentielle Liftung erhalten, sind in [Grö77] zu finden.

Die uns bereits bekannten Kurzzyklus-Ungleichungen können zu den sogenannten **$\mathbf{T_k}$-Ungleichungen** verallgemeinert werden. Diese lassen sich wiederum zu den sogenannten **C2-Ungleichungen** verallgemeinern.

Die genannten Beispiele für asymmetrische Ungleichungen sind für das ATSP-Polytop P_{ATSP} gültig und werden in [GP85] ausführlicher diskutiert. Sie sind hier nur der Vollständigkeit halber erwähnt und werden in unseren Untersuchungen nicht betrachtet, da das Hauptaugenmerk auf solchen Ungleichungen liegt, die die Besonderheit der Zeitfensterrestriktionen bei einem ATSPTW berücksichtigen.

5.4 Die π-, σ- und (π, σ)-Ungleichungen

Bei einem ATSPTW haben wir für jeden Knoten $i \in V$ ein Zeitfenster $[r_i, d_i]$ gegeben. Diese Zeitfenster implizieren auf einfache Weise Reihenfolgebedingungen. Betrachten wir zwei Knoten $i, j \in V$ mit den Zeitfenstern $[r_i, d_i]$ bzw. $[r_j, d_j]$, und sei $\vartheta_{ji} = p_j + c_{ji}$. Für den Fall, daß

$$d_i < r_j + \vartheta_{ji},$$

ist offensichtlich, daß der Knoten i vor dem Knoten j ausgeführt werden muß. Andernfalls würden wir das Zeitfenster des Knotens i verletzen, da die Ausführung des Knotens i nach dem Zeitpunkt d_i beginnen würde. Auf Grund dieser Tatsache können wir also aus den gegebenen Zeitfenstern Reihenfolgebedingungen für die Knoten $i \in V$ ableiten und bekannte Klassen von Ungleichungen, die für das ATSP mit Reihenfolgebedingungen (englisch: ATSP with precedence constraints, kurz: ATSPPC) entwickelt wurden, auch für das ATSPTW verwenden. Das ATSPPC besteht darin, in einem Digraphen $D = (V \cup \{0\}, A)$ einen kostenminimalen Hamiltonkreis zu finden, der im Depotknoten 0 beginnt und endet, so daß er alle gegebenen Reihenfolgebedingungen für die Knoten $i \in V$ erfüllt. Die konvexe Hülle der Inzidenzvektoren von Hamiltonkreisen mit diesen Eigenschaften nennen wir das ATSPPC-Polytop.

Im ersten Unterabschnitt dieses Abschnitts stellen wir die Grundversionen der π-, σ- und (π, σ)-Ungleichungen vor. Im zweiten Unterabschnitt wird eine Liftungs-Prozedur vorgestellt, mit deren Hilfe man diese Klassen von Ungleichungen verallgemeinern kann. Auf Grund der Zeitfenster gibt es eine Menge von Bögen, die in einem zulässigen Hamiltonkreis nicht enthalten sein können. Wird dies berücksichtigt, so lassen sich die π-, σ- und (π, σ)-Ungleichungen verschärfen. Darauf werden wir im dritten Unterabschnitt eingehen. Zuvor benötigen wir jedoch noch einige Begriffe. Im folgenden bezeichnen wir durch

$$i \prec j$$

die Reihenfolgebedingung, daß der Knoten i vor dem Knoten j ausgeführt werden muß. Desweiteren führen wir einen **Präzedenzdigraphen** $P = (V, R)$ ein, wobei $R := \{(i, j) \mid i, j \in V, \, i \prec j\}$. Man beachte, daß P transitiv abgeschlossen ist. Damit ein zulässiger Hamiltonkreis in D existiert, muß P azyklisch sein. Für $S \subseteq V$ bezeichnen wir die Menge der **Vorgänger** von S mit

$$\pi(S) := \{i \in V \mid (i, j) \in R, \, j \in S\}$$

und die **Nachfolger** von S mit

$$\sigma(S) := \{j \in V \mid (i,j) \in R,\, i \in S\}.$$

5.4.1 Die Grundversionen

Die folgenden drei Klassen von Ungleichungen wurden von Balas, Fischetti und Pulleyblank [BFP95] eingeführt.

Satz 5.2
Für $S \subseteq V$, $\overline{S} := V \backslash S$, ist die **Vorgänger-Ungleichung (π-Ungleichung)**

$$x(S\backslash\pi(S) : \overline{S}\backslash\pi(S)) \geq 1$$

gültig bzgl. des ATSPPC-Polytops. □

Satz 5.3
Für $S \subseteq V$, $\overline{S} := V \backslash S$, ist die **Nachfolger-Ungleichung (σ-Ungleichung)**

$$x(\overline{S}\backslash\sigma(S) : S\backslash\sigma(S)) \geq 1$$

gültig bzgl. des ATSPPC-Polytops. □

Satz 5.4
Seien $X, Y \subset V$, so daß $i \prec j$ für jedes Paar $i \in X$, $j \in Y$, und sei $W := \pi(X) \cup \sigma(Y)$. Dann ist für jede Teilmenge $S \subset V$ mit $X \subseteq S$, $Y \subseteq \overline{S}$ die **Vorgänger-Nachfolger-Ungleichung ((π, σ)-Ungleichung)**

$$x(S\backslash W : \overline{S}\backslash W) \geq 1$$

gültig bzgl. des ATSPPC-Polytops. □

Schreibt man für $S \subseteq V$ die uns bekannten SEC $x(A(S)) \leq |S| - 1$ in eine ihnen äquivalente Form

$$x(S : \overline{S}) \geq 1,$$

so sieht man, daß die π-, σ- und (π, σ)-Ungleichungen verschärfte Versionen der SEC sind.

Für die π-, σ- und (π, σ)-Ungleichungen sind derzeit keine polynomialen exakten Separierungsalgorithmen bekannt. Jedoch lassen sich die Ungleichungen abschwächen, indem man in den π-Ungleichungen $\pi(S)$ durch $\pi(j) \subseteq \pi(S)$ für $j \in V$ mit $\pi(j) \neq \emptyset$, in den σ-Ungleichungen $\sigma(S)$ durch $\sigma(j) \subseteq \sigma(V)$ für $j \in V$ mit $\sigma(j) \neq \emptyset$ bzw. in den (π, σ)-Ungleichungen W durch $W_{ij} := \pi(i) \cup \sigma(j)$ für $i, j \in V$ mit $i \prec j$ ersetzt. Wir reden dann von einer **schwachen π-, σ- bzw. (π, σ)-Ungleichung**. Für diese Ungleichungen haben Balas et al. exakte Separierungsalgorithmen entwickelt, die im Rahmen unseres B&C-Algorithmus implementiert sind. Zu deren Beschreibung verweisen wir auf [BFP95] oder auch [Asc95].

5.4.2 Eine Liftungs-Prozedur

Im Abschnitt 5.1 haben wir mit der sequentiellen Liftung bereits eine Liftungs-Methode kennengelernt. In diesem Abschnitt wenden wir uns einer von Balas et al. [BFP95] entwickelten Liftungs-Prozedur zu, die eine Verallgemeinerung der im vorangegangenen Unterabschnitt behandelten Ungleichungen zuläßt und damit zu neuen Klassen von Ungleichungen führt.

In diesem Unterabschnitt bezeichnen wir einen vollständigen Digraphen auf der Knotenmenge V, $|V| = n$, mit $D_n = (V, A_n)$. Es seien ein vollständiger Digraph $D_{n+1} = (V \cup \{0\}, A_{n+1})$ und ein Präzedenzdigraph $P = (V, R)$ mit

$$R := \{(i, j) \,|\, i, j \in V,\ i \prec j\}$$

gegeben. Auf Grund der durch P gegebenen Reihenfolgebedingungen können bestimmte Bögen aus A_{n+1} in keinem Hamiltonkreis, der alle Reihenfolgebedingungen erfüllen soll, enthalten sein. Das folgende Lemma gibt einige Bedingungen für derartige Bögen an.

Lemma 5.5
Es sei $D_{n+1} = (V \cup \{0\}, A_{n+1})$ ein vollständiger Digraph. Ein Bogen $(i, j) \in A_{n+1}$ ist für einen Hamiltonkreis, der alle durch $P = (V, R)$ gegebenen Reihenfolgebedingungen erfüllen soll, **unzulässig**, *falls mindestens eine der folgenden Bedingungen zutrifft:*

(i) $i = 0,\ j \in \sigma(V)$,

(ii) $i \in \pi(V),\ j = 0$,

(iii) $(j, i) \in R$,

(iv) $\exists k \in V$ *mit* $(i, k) \in R$ *und* $(k, j) \in R$.

Beweis:
Offensichtlich.

□

Durch das Entfernen aller Bögen $(i, j) \in A_{n+1}$, die mindestens eine der Bedingungen aus Lemma 5.5 erfüllen, erhalten wir aus $D_{n+1} = (V \cup \{0\}, A_{n+1})$ den Digraphen $D = (V \cup \{0\}, A)$.

Betrachten wir nun eine Zerlegung der Knotenmenge $V \cup \{0\}$ in $m \leq n$ nichtleere Teilmengen $S_1, \ldots, S_m$ mit $S_1 = \{0\}$ und ersetzen jede Menge S_u durch einen einzigen Knoten u für $u = 1, \ldots, m$, so erhalten wir die Knotenmenge V^* mit $|V^*| = m$. Die Funktion $h : \mathbb{N} \to \mathbb{N}$ gebe für jeden Knoten $i \in V \cup \{0\}$ den Index $h(i) = u$ der Knotenmenge S_u mit $i \in S_u$ an. Es sei $D^*_m = (V^*, A^*_m)$ der vollständige Digraph auf $m = |V^*|$ Knoten. Der zu D^* gehörende Präzedenzdigraph sei durch $P^* = (V^*, R^*)$ mit

$$R^* := \{(u,v)\,|\,u,v \in V^*,\ u \prec v\}$$

gegeben, wobei $u \prec v$, falls Knoten $i \in S_u$ und $j \in S_v$ existieren, so daß $i \prec j$. Man beachte, daß P^* transitiv abgeschlossen ist und Kreise enthalten kann, obwohl P azyklisch ist. Wir betrachten im folgenden nur Zerlegungen der Knotenmenge $V \cup \{0\}$, bei denen der resultierende Präzedenzdigaph P^* azyklisch ist. Es sei

$$D^* = (V^*, A^*)$$

der Digraph, den wir aus $D^*_m = (V^*, A^*_m)$ durch das Entfernen der unzulässigen Bögen $(u,v) \in A^*_m$ entsprechend Lemma 5.5 erhalten. Wir bezeichnen mit $P(D)$ das ATSPPC-Polytop bzgl. $D = (V \cup \{0\}, A)$ und mit $P(D^*)$ das ATSPPC-Polytop bzgl. $D^* = (V^*, A^*)$. Wir definieren die Bogenmengen A_0 und A_1 durch

$$\begin{aligned} A_0 &:= \{(i,j) \in A\,|\,(h(i), h(j)) \in A^*\}, \\ A_1 &:= \{(i,j) \in A\,|\,(h(i), h(j)) \notin A^*\}. \end{aligned}$$

Wir können nun die Liftungs-Prozedur für die π-, σ- und (π,σ)-Ungleichungen wie folgt formulieren. Zur Vereinfachung der Notation schreiben wir im folgenden $\beta x \geq \beta_0$ anstelle $\beta^T x \geq \beta_0$.

Satz 5.5
Es sei $\beta^* y \geq \beta_0^*$ eine gültige Ungleichung für $P(D^*)$. Es seien τ_{ij} für $(i,j) \in A_1$ beliebige endliche Zahlen. Dann existieren Zahlen ν_h^0 für $h = 1, \ldots, m$ mit $\nu_h^0 = 0$, falls $|S_h| = 1$, so daß für beliebige $\nu_h \geq \nu_h^0$, $h = 1, \ldots, m$, die Ungleichung $\beta x \geq \beta_0$ mit

$$\beta_{ij} := \begin{cases} 0 & \text{falls } h(i) = h(j) \text{ und } (i,j) \in A \\ \beta^*_{h(i)\,h(j)} + \nu_{h(i)} & \text{falls } h(i) \neq h(j) \text{ und } (i,j) \in A_0 \\ \tau_{ij} + \nu_{h(i)} & \text{falls } h(i) \neq h(j) \text{ und } (i,j) \in A_1 \end{cases}$$

und

$$\beta_0 := \beta_0^* + \sum_{h=1}^{m} \nu_h$$

gültig bzgl. $P(D)$ ist.

Beweis:
Siehe [BFP95].

□

Am Beispiel der π-Ungleichungen wollen wir zeigen, wie diese sich durch Anwendung der Liftungs-Prozedur verallgemeinern lassen. Für eine beliebige Teilmenge $S^* \subset V^*$ bezeichnen wir mit $\pi^*(S^*)$ die Menge der Vorgänger von S^* bzgl. des Präzedenzdigraphen P^*. Dann erhalten wir folgendes Resultat.

Satz 5.6
Für eine beliebige Teilmenge $S^* \subset V^*$ seien $W := \bigcup_{h \in S^*} S_h$ und $P_W := \bigcup_{h \in \pi^*(S^*)} S_h$. Dann ist die **verallgemeinerte π-Ungleichung**

$$x(W \backslash P_W : \overline{W} \backslash P_W) + \sum_{\substack{h \in S^* \\ |S_h| \geq 2}} (x(S_h : \overline{S}_h) - 1) \geq 1$$

gültig bzgl. $P(D)$.

Beweis:
Siehe [BFP95].

□

In [BFP95] ist ein heuristischer Separierungsalgorithmus für die verallgemeinerten π-Ungleichungen skizziert. Sei $\beta^* y \geq \beta_0^*$ eine gültige Ungleichung für $P(D^*)$ und $\beta x \geq \beta$ die korrespondierende gültige Ungleichung für $P(D)$. Für ein $\bar{x} \in \mathbb{R}^A$ mit $\bar{x}(A(S_h)) = |S_h| - 1,\ h = 1, \ldots, m$ definieren wir

$$\bar{y}_{uv} := \sum_{\substack{i \in S_u \\ j \in S_v}} \bar{x}_{ij}$$

für alle $u, v \in V^*$. Finden wir nun eine bzgl. $P(D^*)$ gültige Ungleichung $\beta^* y \geq \beta_0^*$, die von $\bar{y}$ verletzt wird, so haben wir auch eine bzgl. $P(D)$ gültige Ungleichung $\beta x \geq \beta$ gefunden, die von $\bar{x}$ verletzt wird, d.h., $\beta^* \bar{y} < \beta_0^*$ impliziert $\beta \bar{x} < \beta_0$. Die Frage, wie man die Knotenmenge $V \cup \{0\}$ in nichtleere Teilmengen $S_1, \ldots, S_m$ mit $S_1 = \{0\}$ und $i \in S_{h(i)}$ für $i \in V$ zerlegt, so daß die Kurzzyklus-Ungleichungen

$$x(A(S_h)) \leq |S_h| - 1 \text{ für } h = 1, \ldots, m$$

mit Gleichheit erfüllt sind, ist in [BFP95] allerdings offen gelassen.

Wir haben für das Schnittebenenverfahren einen heuristischen Separierungsalgorithmus für die verallgemeinerten π-Ungleichungen implementiert, der auf folgender Idee beruht. Es sei $\bar{x} \notin P(D)$ mit $0 \leq \bar{x}_{ij} \leq 1$ für alle $(i, j) \in A$ ein gegebener Punkt in $\mathbb{R}^A$, der die Gradbeschränkungen erfüllt. Desweiteren sei $D(\bar{x}) = (V \cup \{0\}, A(\bar{x}))$ der durch die Bogenmenge $(i, j) \in A$ mit $\bar{x}_{ij} > 0$ induzierte Unterdigraph von D. Wir bestimmen nun in $D(\bar{x})$ gerichtete Wege, die aus Bögen $(i, j) \in A(\bar{x})$ mit $\bar{x}_{ij} = 1$ (sogenannte **1-Bögen**) bestehen. Solche Wege nennen wir **1-Wege**. Jetzt zerlegen wir die Knotenmenge $V \cup \{0\}$ in die Mengen $S_1, \ldots S_m$, so daß $S_1 = \{0\}$, jede Menge S_h mit $|S_h| = 1$ genau einem Knoten in V und jede Menge S_h mit $|S_h| \geq 2$ genau der Knotenmenge eines 1-Weges in $D(\bar{x})$ entspricht. Damit ist garantiert, daß $x(A(S_h)) = |S_h| - 1$ für $h = 1, \ldots, m$. Wir schrumpfen nun die 1-Bögen $(i, j) \in A(\overline{x})$ wie folgt zusammen:

- verschmelze die Knoten i und j zu einem Knoten h,

- ersetze die Bögen $(k,i) \in A(\bar{x})$ bzw. die Bögen $(j,l) \in A(\bar{x})$ durch die Bögen (k,h) bzw. (h,l) für $k,l \in V$.

Damit erhalten wir den Digraphen $D^* = (V^*, A^*)$. Durch das iterative Zusammenschrumpfen der 1-Bögen wird jeder 1-Weg durch einen Knoten $h \in V^*$ ersetzt. Den Präzedenzdigraphen $P^* = (V^*, R^*)$ erhalten wir auf folgende Weise. Für jeden 1-Bogen $(i,j) \in A(\bar{x})$ verschmelzen wir die korrespondierenden Knoten i und j in dem Digraphen P wie folgt:

- verschmelze die Knoten i und j zu einem Knoten h,
- ersetze jedes Paar von Bögen $(i,k) \in R$ und $(j,k) \in R$ bzw. $(l,i) \in R$ und $(l,j) \in R$ durch einen Bogen (h,k) bzw. (l,h) für $k,l \in V$,
- ersetze alle anderen Bögen in R, die i bzw. j als Anfangs- oder Endknoten haben, durch den entsprechenden Bogen, der h als Anfangs- oder Endknoten hat.

Wir formulieren nun einen heuristischen Separierungsalgorithmus für die verallgemeinerten π-Ungleichungen. Die Grundidee liegt, wie oben bereits erläutert, in dem Zusammenschrumpfen der 1-Wege. Für die Implementierung wird der Depotknoten 0 in einen Startknoten s und einen Endknoten d aufgespalten, so daß jede Tour im Knoten s beginnt und im Knoten d endet.

Heuristischer Separierungsalgorithmus für die verallgemeinerten π-Ungleichungen.

Input: $D = (V \cup \{s,d\}, A)$, $P = (V,R)$ und eine gebrochene LP-Lösung $\bar{x} \in \mathbb{R}^A$, welche die Gradbeschränkungen erfüllt.

Output: Verallgemeinerte π-Ungleichungen, die von $\bar{x}$ verletzt werden, sofern heuristisch welche gefunden werden können.

1. Konstruiere den Digraphen $D(\bar{x}) = (V \cup \{s,d\}, A(\bar{x}))$.
2. Bestimme die 1-Wege in $D(\bar{x})$.
3. Zerlege die Knotenmenge $V \cup \{0\}$ in die Mengen $S_1, \dots S_m$, so daß $S_1 = \{s\}$, jede Menge S_h mit $|S_h| = 1$ genau einem Knoten in V und jede Menge S_h mit $|S_h| \geq 2$ genau einer Knotenmenge eines 1-Weges in $D(\bar{x})$ entspricht.
4. Setze $\bar{y}_{uv} := \sum_{\substack{i \in S_u \\ j \in S_v}} \bar{x}_{ij}$ und konstruiere den Digraphen $D^*(\bar{y}) = (V^*, A^*(\bar{y}))$ durch das Zusammenschrumpfen der 1-Wege sowie den dazugehörigen Präzedenzdigraphen $P^* = (V^*, R^*)$ durch das Verschmelzen der korrespondierenden Knoten in $P = (V,R)$.

5. Für alle Knoten $v \in V^*\backslash\{h(d)\}$ mit $\pi^*(v) \neq \emptyset$:

   5.1 Konstruiere einen Hilfsdigraphen $D' = (V', A')$ durch das Entfernen der Knoten in $\pi^*(v)$ und der inzidenten Bögen aus $D^*(\bar{y})$, d.h., $V' = V^*\backslash\pi^*(v)$ und $A' = \{(u,v) \in A^*(\bar{y}) \,|\, u, v \in V'\}$.

   5.2 Setze die Bogengewichte $c'_{uv} := \bar{y}_{uv}\ \forall (u,v) \in A'$.

   5.3 Berechne einen minimalen $(v, h(d))$-Schnitt $\delta^+(S^*)$ in $D' = (V', A')$.

   5.4 Falls $c'(\delta^+(S^*)) < 1$, setze $W := \bigcup_{h \in S^*} S_h$ und $P_W := \bigcup_{h \in \pi^*(S^*)} S_h$ und gebe die verallgemeinerte π-Ungleichung

$$x(W\backslash P_W : \overline{W}\backslash P_W) + \sum_{\substack{h \in S^* \\ |S_h| \geq 2}} (x(S_h : \overline{S}_h) - 1) \geq 1,$$

   die von $\bar{x}$ verletzt wird, aus.

6. Stop.

Es sei angemerkt, daß die Schritte 5.1 bis 5.3 einem exakten Separierungsalgorithmus für die schwachen π-Ungleichungen entsprechen. Finden wir eine solche schwache π-Ungleichung

$$y(S^*\backslash\pi^*(v) : \overline{S}^*\backslash\pi^*(v)) \geq 1$$

für ein $v \in V^*\backslash\{h(d)\}$, die von $\bar{y}$ verletzt wird, und ersetzen $\pi^*(v)$ durch $\pi^*(S^*)$ mit $S^* \subset V^*\backslash\{h(d)\}$ und $v \in S^*$, so wird die (schärfere) π-Ungleichung

$$y(S^*\backslash\pi^*(S^*) : \overline{S}^*\backslash\pi^*(S^*)) \geq 1$$

offensichtlich ebenfalls, und sogar im stärkeren Maße, von $\bar{y}$ verletzt. Aus dieser π-Ungleichung erhalten wir dann im Schritt 5.4 durch Anwendung der Liftungs-Prozedur entsprechend dem Satz 5.6 eine verallgemeinerte π-Ungleichung, die von $\bar{x}$ verletzt wird.

Einen minimalen $(v, h(d))$-Schnitt im Schritt 5.3 können wir auf Grund des Max-Flow Min-Cut Theorems durch die Berechnung eines maximalen $(v, h(d))$-Flusses effizient bestimmen (siehe hierzu [Grö94]). Mit einer Laufzeit von $O(n^3)$ für die Berechnung des maximalen Flusses hat der Separierungsalgorithmus eine Laufzeit von $O(n^4)$.

Wir hatten bereits erwähnt, daß der Präzedenzdigraph $P^* = (V^*, R^*)$, den wir durch das Verschmelzen der Knoten i und j mit $\bar{x}_{ij} = 1$ aus $P = (V, R)$ erhalten, nicht notwendiger Weise azyklisch zu sein braucht. Um auszuschließen, daß der Präzedenzdigraph P^* in dem Separierungsalgorithmus für die verallgemeinerten π-Ungleichungen Kreise enthält, werden wir vorher einen Separierungsalgorithmus (siehe [BFP95]) aufrufen, der im Falle des Vorhandenseins verletzte Ungleichungen bzgl. gerichteter Kreise in dem Präzedenzdigraphen P^* (englisch: precedence cycle breaking inequalities, kurz: PCB-Ungleichungen) findet.

Satz 5.7
Seien $S_1, \dots, S_m \subset V$, $m \geq 2$, disjunkte Knotenmengen, so daß $S_i \cap \pi(S_{i+1}) \neq \emptyset$ für $i = 1, \dots, m$ mit $S_{m+1} = S_1$. Dann ist die **PCB-Ungleichung**

$$\sum_{i=1}^{m} x(A(S_i)) \leq \sum_{i=1}^{m} |S_i| - m - 1$$

gültig bzgl. des ATSPPC-Polytops.

Beweis:
Siehe [BFP95].

$\square$

Da der Separierungsalgorithmus für die PCB-Ungleichungen auf dem Zusammenschrumpfen (englisch: shrinking) der 1-Wege beruht, nennen wir ihn auch Shrinking-Separierungsalgorithmus. Dieser Separierungsalgorithmus findet neben verletzten PCB-Ungleichungen auch verletzte Ungleichungen, die wir auf Grund des Lemmas 5.5 formulieren können (siehe [BFP95]). Diese Ungleichungen nennen wir entsprechend der Implementierung von Ascheuer **Spezial-Ungleichungen** .

5.4.3 Verschärfte π-, σ- und (π, σ)-Ungleichungen

Im Unterabschnitt 5.4.1 haben wir die π-, σ- und (π, σ)-Ungleichungen eingeführt. Diese lassen sich für das ATSPTW verschärfen, da auf Grund der gegebenen Zeitfenster bestimmte Bögen des vollständigen Digraphen $D = (V \cup \{0\}, A)$ in einem zulässigen Hamiltonkreis nicht enthalten sein können. Derartige Bögen würden zu unzulässigen Wegen führen (siehe Unterabschnitt 4.4.1). In [Asc95] wurden die verschärften π-, σ- und (π, σ)-Ungleichungen eingeführt. Die verschärften π- und σ-Ungleichungen erwiesen sich jedoch für das ATSPTW als ungeeignet. Die Separierungsalgorithmen für diese Ungleichungen erfordern zwei fest vorgegebene Knoten s bzw. d als Start- bzw. Endknoten. Die Zeitfenster dieser beiden Knoten so gewählt sind, daß jeder zulässige Hamiltonkreis im Knoten s beginnt und im Knoten d endet. Für die verschärften π-Ungleichungen war damit für eine beliebige Teilmenge $S \subset V \backslash \{d\}$ die Menge der bzgl. der Zeitfenster unzulässigen Bögen $Q = Q_1 \cup Q_2$ für die von uns getesteten Problembeispielen leer, wobei

$$Q_1 := \{(i,k), (k,d) \in A \,|\, \exists i \in S, \text{ so daß der Weg } (i,k,d) \text{ unzulässig ist}\},$$
$$Q_2 := \{(u,v) \in A \,|\, \exists i \in S, \text{ so daß der Weg } (i,u,v,d) \text{ unzulässig ist}\}.$$

Ein Weg $(v_1, \dots, v_k)$ ist unzulässig, falls $r_{v_1} + \vartheta_{v_1 v_2} + \ldots + \vartheta_{v_{k-1} v_k} > d_{v_k}$. Ist nun d_{v_k} sehr groß gewählt, wie im Fall des Endknotens d, so ist Q offensichtlich leer. Analoges gilt auch für die σ-Ungleichungen. Aus diesem Grund beschränken wir uns hier auf die verschärften (π, σ)-Ungleichungen.
Lemma 5.6
Seien $D = (V \cup \{0\}, A)$ und $P = (V, R)$ gegeben, und seien $X, Y \subset V$, so daß $i \prec j$ für jedes Paar $i \in X, j \in Y$. Definiere $W := \pi(X) \cup \sigma(Y)$. Es gelte die

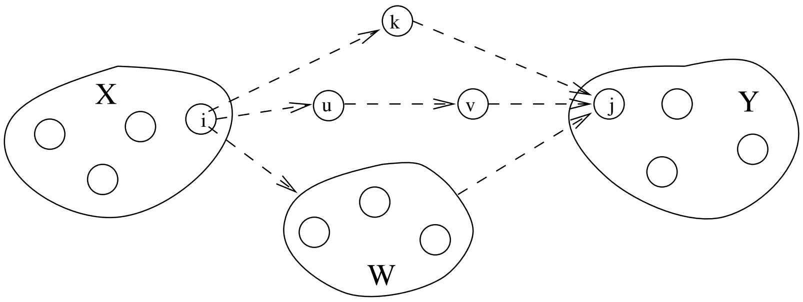

Abbildung 5.2 Verschärfte (π, σ)-Ungleichungen

Dreiecksungleichung $\vartheta_{ij} \leq \vartheta_{ik} + \vartheta_{kj}$ *für alle* $i \neq j \neq k \neq i$. *Sei* $Q = Q_1 \cup Q_2$ *die Menge der unzulässigen Bögen, wobei*

$$Q_1 := \{(i,k), (k,j) \in A \,|\, \exists i \in X, j \in Y \text{ so daß der Weg } (i,k,j) \text{ unzulässig ist}\},$$
$$Q_2 := \{(u,v) \in A \,|\, \exists i \in X, j \in Y \text{ so daß der Weg } (i,u,v,j) \text{ unzulässig ist}\},$$

und sei $S \subset V$ *mit* $X \subset S$ *und* $Y \subset \overline{S}$. *Dann ist die* **verschärfte (π, σ)-Ungleichung**

$$x((S \backslash W : \overline{S} \backslash W) \backslash Q) \geq 1$$

gültig bzgl. $P_1^{TW}, P_2^{TW}, P_3^{TW}$.

Beweis:
Ein zulässiger Hamiltonkreis, der über einen Knoten $i \in X$ die Knotenmenge X verläßt und über einen Knoten $j \in Y$ in die Knotenmenge Y hineingeht, kann dazwischen keinen Knoten aus $W := \pi(X) \cup \sigma(Y)$ durchlaufen, ohne daß Reihenfolgebedingungen und damit Zeitfenster verletzt werden würden. Darüber hinaus darf er beim Übergang von der Knotenmenge X zur Knotenmenge Y auf Grund der Dreiecksungleichung keinen Bogen aus Q_1 oder Q_2 durchlaufen, da er sonst Zeitfenster verletzen würde. Daraus folgt, daß jeder zulässige Hamiltonkreis wenigstens einen Bogen aus der Bogenmenge $(S \backslash W : \overline{S} \backslash W) \backslash Q$ durchlaufen muß. Die Abbildung 5.2 stellt diese Überlegung schematisch dar.

□

Für die verschärften (π, σ)-Ungleichungen ist kein polynomialer exakter Separierungsalgorithmus bekannt. In Anlehnung an einen exakten Separierungsalgorithmus für die schwachen (π, σ)-Ungleichungen (siehe [Asc95]) können wir jedoch einen exakten Separierungsalgorithmus für eine vereinfachte Version der verschärften (π, σ)-Ungleichungen mit $X = \{i\}$ und $Y = \{j\}$ für ein Knotenpaar $i, j \in V$ mit $i \prec j$ formulieren.

Exakter Separierungsalgorithmus für die vereinfachte Version der verschärften (π, σ)-Ungleichungen.

Input: $D = (V \cup \{s, d\}, A)$, $P = (V, R)$ und eine gebrochene LP-Lösung $\bar{x} \in \mathbb{R}^A$, welche die Gradbeschränkungen erfüllt.

Output: Verschärfte (π, σ)-Ungleichungen, die von $\bar{x}$ verletzt werden, oder die Antwort, daß keine solchen Ungleichungen von $\bar{x}$ verletzt werden.

1. Konstruiere den durch die Bogenmenge $(i, j) \in A$ mit $\bar{x}_{ij} > 0$ induzierten Unterdigraphen $D(\bar{x}) = (V \cup \{s, d\}, A(\bar{x}))$.

2. Für alle Bögen $(i, j) \in R$:

   2.1 Konstruiere einen Hilfsdigraphen $D' = (V', A')$ durch das Entfernen der Knoten in $\pi(i) \cup \sigma(j)$ und aller Knoten k mit $r_i + \vartheta_{ik} + \vartheta_{kj} > d_j$ sowie aller entsprechenden inzidenten Bögen und aller Bögen (u, v) mit $r_i + \vartheta_{iu} + \vartheta_{uv} + \vartheta_{vj} > d_j$ aus $D(\bar{x})$.

   2.2 Setze die Bogengewichte $c'_{ij} := \bar{x}_{ij}\ \forall (i, j) \in A'$.

   2.3 Berechne einen minimalen (i, j)-Schnitt $\delta^+(S)$ in $D' = (V', A')$.

   2.4 Falls $c'(\delta^+(S)) < 1$, setze $W_{ij} := \pi(i) \cup \sigma(j)$ und $Q = Q_1 \cup Q_2$, wobei

$$\begin{aligned} Q_1 &:= \{(i,k),\, (k,j) \in A(\bar{x}) \,|\, r_i + \vartheta_{ik} + \vartheta_{kj} > d_j\}, \\ Q_2 &:= \{(u,v) \in A(\bar{x}) \,|\, r_i + \vartheta_{iu} + \vartheta_{uv} + \vartheta_{vj} > d_j\}, \end{aligned}$$

   und gebe die Ungleichung

$$x((S \backslash W_{ij} : \overline{S} \backslash W_{ij}) \backslash Q) \geq 1,$$

   die von $\bar{x}$ verletzt wird, aus.

3. Stop.

Mit einer Laufzeit von $O(n^3)$ für die Berechnung eines maximalen (i, j)-Flusses im Schritt 2.3 hat der Separierungsalgorithmus eine Laufzeit von $O(|R| \cdot n^3)$. Man beachte, daß der Schritt 2 nur für solche Bögen $(i, j) \in R$ ausgeführt zu werden braucht, die nicht aus der Transitivität der Reihenfolgebedingungen hervorgegangen sind.

5.5 Unzulässige-Wege-Beschränkungen

Die Unzulässige-Wege-Beschränkungen haben wir bereits im Unterabschnitt 4.4.1 im Zusammenhang mit der Modellierung 1 kennengelernt. Sie wurden ursprünglich von Ascheuer [Asc95] zur Modellierung des AHPPTW entwickelt. Diese Ungleichungen beruhen auf der Überlegung, daß bestimmte Wege in einem zulässigen Hamiltonkreis nicht enthalten sein können, da sonst Zeitfensterrestriktionen verletzt werden würden. Wir hatten solche Wege als **unzulässig** bezeichnet.

Die Grundversion der Unzulässige-Wege-Beschränkungen ist durch

$$x(P) \leq k - 2$$

für alle unzulässigen Wege $P = (v_1, \ldots, v_k)$ gegeben. Es gibt verschiedene Möglichkeiten der Liftung dieser Ungleichung. Von Ascheuer wurden Separierungsalgorithmen für die verschiedenen Versionen der Unzulässige-Wege-Beschränkungen implementiert, die wir in unserem B&C-Algorithmus verwenden. Aus diesem Grund seien sie an dieser Stelle erwähnt. Wir beschränken uns jedoch auf die Beschreibung der sogenannten Turnier-Ungleichungen und verweisen ansonsten auf [Asc95].

Abbildung 5.3

Die Abbildung 5.3 zeigt, wie wir aus der Ungleichung $x(P) \leq k-2$ eine Turnier-Ungleichung erhalten. Wir erweitern einen unzulässigen Weg $P = (v_1, \ldots, v_k)$ durch das Hinzunehmen aller transitiven Bögen, so daß die Knoten $v_1, \ldots, v_k$ transitiv abgeschlossen sind. Die **Turnier-Ungleichung** ist dann durch

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} x_{v_i v_j} \leq k - 2$$

gegeben. Für diese Ungleichungen gibt es verschiedene geliftete Versionen, zu deren Beschreibung wir auf [Asc95] verweisen. Ebenfalls dort nachzulesen ist, daß das Zusammenfügen zulässiger Wege zu Unzulässigkeiten führen kann, woraus neue Ungleichungen abgeleitet werden können.

5.6 Ungleichungen mit Zeitvariablen

Die Modellierungen 2 und 3 für das ATSPTW unterscheiden sich von der Modellierung 1 vor allem durch das Vorhandensein von Zeitvariablen. In diesem Abschnitt wollen wir für die Modellierung 2 verschärfte Versionen für die MTZ-Ungleichungen, geliftete Zeitfensterrestriktionen (englisch: lifted t-bounds) und die sogenannten 2-Job-Ungleichungen (englisch: two-job-cuts) vorstellen. Diese

Ungleichungen werden wir für die Modellierung 2 hinsichtlich ihrer Auswirkungen auf unseren B&C-Algorithmus untersuchen. Für die Modellierung 3 erweist sich die enge Verknüpfung der x- und t-Variablen an dieser Stelle als Nachteil, da der Versuch, gültige Ungleichungen bzgl. der Zeitvariablen für das Polytop P_3^{TW} herzuleiten, oft zu quadratischen Ungleichungen führt. In [vE95] wurden derartige quadratische Ungleichungen linearisiert, um sie in einem Schnittebenenverfahren zur Lösung des DMPTW (Delivery Man Problem mit Zeitfenstern) zu verwenden. Jedoch erwiesen sie sich für das DMPTW als sehr schwach, so daß wir von deren Implementierung in unserem B&C-Algorithmus Abstand genommen haben.

5.6.1 Verschärfte MTZ-Ungleichungen

Im Unterabschnitt 4.4.2 haben wir die MTZ-Ungleichungen

$$t_i + \vartheta_{ij} - (1 - x_{ij})M \leq t_j$$

kennengelernt. Desrochers und Laporte [DL91] haben verschiedene geliftete Versionen der MTZ-Ungleichungen formuliert. Wir betrachten hier die bzgl. Rückwärtsbögen gelifteten MTZ-Ungleichungen. Es sei $D = (V \cup \{0\}, A)$ ein vollständiger Digraph. Betrachten wir die MTZ-Ungleichungen für einen Bogen $(i,j) \in A$ mit $i,j \neq 0$, so kann diese für den Rückwärtsbogen $(j,i) \in A$ geliftet werden. Wir gehen davon aus, daß beide Bögen bzgl. der Zeitfenster zulässig sind.

Lemma 5.7
Seien $i,j \in V$, $a_{ji} := \max\{\vartheta_{ji}, r_i - d_j\}$ und $M_{ij} \geq d_i + \vartheta_{ij} - r_j$. Dann ist die geliftete MTZ-Ungleichung

$$t_i + \vartheta_{ij} - (1 - x_{ij})M_{ij} + (M_{ij} - \vartheta_{ij} - a_{ji})x_{ji} \leq t_j$$

gültig bzgl. P_2^{TW}.

Beweis:
Sei α_{ji} der Liftungskoeffizient der Variablen x_{ji}, d.h.

$$t_i + \vartheta_{ij} - (1 - x_{ij})M_{ij} + \alpha_{ji}x_{ji} \leq t_j.$$

Ist $x_{ji} = 0$, so ist α_{ji} beliebig. Sei also $x_{ji} = 1$. Dann ist $x_{ij} = 0$, d.h.

$$t_i + \vartheta_{ij} - M_{ij} + \alpha_{ji} \leq t_j.$$

Um α_{ji} zu bestimmen, haben wir 3 Fälle zu untersuchen.

<u>Fall 1:</u> $r_j + \vartheta_{ji} \geq r_i$.
Es gilt $t_i = t_j + \vartheta_{ji}$. Man man beachte, daß $r_j + \vartheta_{ji} \leq d_i$, da sonst (j,i) unzulässig wäre. Daraus folgt, daß

$$\alpha_{ji} \leq M_{ij} - \vartheta_{ij} - \vartheta_{ji}.$$

Fall 2: $d_j + \vartheta_{ji} < r_i$.
Es gilt $t_i = r_i$, d.h.

$$r_i + \vartheta_{ij} - M_{ij} + \alpha_{ji} \leq t_j \leq d_j,$$

und daraus folgt

$$\alpha_{ji} \leq M_{ij} - \vartheta_{ij} - (r_i - d_j).$$

Fall 3: $r_j + \vartheta_{ji} < r_i$ und $d_j + \vartheta_{ji} \geq r_i$.
Es gilt entweder $t_i = r_i$ (Fall 2) oder $t_i = t_j + \vartheta_{ji}$ (Fall 1).

Somit gilt für alle 3 Fälle die Ungleichung

$$\alpha_{ji} \leq M_{ij} - \vartheta_{ij} - \max\{\vartheta_{ji}, r_i - d_j\}.$$

□

Eine weitere Möglichkeit der Verschärfung der MTZ-Ungleichungen wurde von Ascheuer in [Asc95] angegeben. Die MTZ-Ungleichungen berücksichtigen in ihrer Grundversion nur den Fall, daß der Knoten j unmittelbar nach dem Knoten i besucht wird. Betrachten wir nun zusätzlich die durch die Zeitfenster implizierten Reihenfolgebedingungen, so lassen sich die MTZ-Ungleichungen wie folgt verschärfen. Für zwei Knoten $k, l \in V$ sei ω_{kl} das Gewicht des kürzesten Weges von k nach l bzgl. der Gewichte ϑ_{ij} für alle $(i,j) \in A$.

Lemma 5.8
Sei $(i,j) \in A$, $j \neq 0$, mit $i \prec j$. Dann ist die verschärfte MTZ-Ungleichung

$$t_i + \vartheta_{ij} x_{ij} + \sum_{\substack{k \in V \\ k \neq i,j}} (\omega_{ik} + \vartheta_{kj}) x_{kj} \leq t_j$$

gültig bzgl. P_2^{TW}.

Beweis:
Da $i \prec j$, muß $t_i \leq t_j$ gelten. Darüber hinaus ist

$$t_i + \vartheta_{ij} x_{ij} \leq t_j$$

eine gültige Ungleichung. Man beachte, daß diese Ungleichung in trivialer Weise erfüllt ist, falls $d_i + \vartheta_{ij} \leq r_j$, da

$$t_i + \vartheta_{ij} x_{ij} \leq d_i + \vartheta_{ij} x_{ij} \leq d_i + \vartheta_{ij} \leq r_j \leq t_j.$$

Da j nicht notwendiger Weise unmittelbar nach i ausgeführt werden muß, können Knoten zwischen den Knoten i und j besucht werden. Nehmen wir an, auf dem

Weg von i nach j wird ein Knoten $k \in V$ unmittelbar vor dem Knoten j ausgeführt. Dann ist

$$t_i + \vartheta_{ij} x_{ij} + (\omega_{ik} + \vartheta_{kj}) x_{kj} \leq t_j$$

eine gültige Ungleichung, die auf Grund der geforderten Gradbeschränkungen $\sum_{j \in V} x_{kj} = \sum_{k \in V} x_{kj} = 1$ zu

$$t_i + \vartheta_{ij} x_{ij} + \sum_{\substack{k \in V \\ k \neq i,j}} (\omega_{ik} + \vartheta_{kj}) x_{kj} \leq t_j$$

verschärft werden kann.

□

Analog erhalten wir eine gültige Ungleichung

$$t_i + \vartheta_{ij} x_{ij} + \sum_{\substack{k \in V \\ k \neq i,j}} (\vartheta_{ik} + \omega_{kj}) x_{kj} \leq t_j,$$

wenn wir den Fall betrachten, daß auf dem Weg von i nach j der Knoten k unmittelbar nach Knoten i besucht wird.

5.6.2 Geliftete Zeitfensterrestriktionen

Die Zeitfensterrestriktionen $r_i \leq t_i \leq d_i$ geben uns eine untere und obere Schranke für die Zeitvariablen t_i in der Modellierung 2 an. Desrochers und Laporte [DL91] haben folgende geliftete Version der unteren und oberen Schranken für die Zeitvariablen angegeben. Wir betrachten wiederum nur bzgl. der Zeitfenster zulässige Bögen (i, j), d.h., $r_i + \vartheta_{ij} \leq d_j$. Für den Depotknoten gelte $r_0 = d_0 = 0$, so daß $t_0 = 0$ ist.

Lemma 5.9
Seien $a_{ji} := max\{0, r_j - r_i + \vartheta_{ji}\}$ *und* $b_{ij} := max\{0, d_i - d_j + \vartheta_{ij}\}$, $b_{i0} = 0$. *Dann sind die Ungleichungen* (**lifted t-bounds**)

$$\text{(i)} \quad r_i + \sum_{\substack{j=0 \\ j \neq i}}^{n} a_{ji} x_{ji} \leq t_i \quad \forall i \in V,$$

$$\text{(ii)} \quad d_i - \sum_{\substack{j=0 \\ j \neq i}}^{n} b_{ij} x_{ij} \geq t_i \quad \forall i \in V$$

gültig bzgl. P_2^{TW}.

Beweis:
(i) Sei $i \in V$ und sei a_{ji} der Liftungskoeffizient der Variablen x_{ji}, d.h.

$$r_i + a_{ji}x_{ji} \leq t_i.$$

Ist $x_{ji} = 0$, so ist a_{ji} beliebig. Sei also $x_{ji} = 1$, d.h., der Knoten j wird unmittelbar vor dem Knoten i besucht, und es gilt

$$r_i + a_{ji} \leq t_i.$$

Die frühestmögliche Startzeit für die Ausführung des Knotens i ist $t_i = r_i$ bzw. $t_i = r_j + \vartheta_{ji}$. Damit folgt

$$a_{ji} \leq 0$$

bzw.

$$a_{ji} \leq r_j - r_i + \vartheta_{ji}.$$

Falls $r_j + \vartheta_{ji} \geq r_i$, so wählen wir $a_{ji} = r_j - r_i + \vartheta_{ji}$ (man beachte, daß $r_j + \vartheta_{ji} \leq d_i$, da (j, i) bzgl. der Zeitfenster zulässig ist). Andernfalls wählen wir $a_{ji} = 0$, da sonst a_{ji} negativ wäre, was im Widerspruch zu $t_i \geq r_i$ stehen würde.
Mit $a_{ji} := \max\{0, r_j - r_i + \vartheta_{ji}\}$ ist dann

$$r_i + a_{ji}x_{ji} \leq t_i$$

eine gültige Ungleichung bzgl. des Polytops P_2^{TW}. Auf Grund der Gradbeschränkungen $\sum_{j=0}^{n} x_{ji} = 1$ können wir diese Ungleichung zu

$$r_i + \sum_{\substack{j=0 \\ j \neq i}}^{n} a_{ji}x_{ji} \leq t_i$$

verschärfen.
(ii) Analog zu (i).

$\square$

5.6.3 2-Job-Ungleichungen

Das ATSPTW läßt sich auf einfache Weise als ein 1-Maschinen-Scheduling Problem interpretieren. Mit jedem Knoten $i \in V$ assoziieren wir einen Auftrag i, für den durch $[r_i, d_i]$ ein Zeitfenster gegeben ist. Mit der Ausführung des Auftrags i kann frühestens zum Zeitpunkt r_i bzw. muß spätestens zum Zeitpunkt d_i begonnen werden. Mit p_i bezeichnen wir die Ausführungszeit des Auftrags i

und mit c_{ij} die benötigte Zeit für das Umrüsten der Maschine, um Auftrag j unmittelbar nach Auftrag i auszuführen. Die Werte $\vartheta_{ij} := p_i + c_{ij}$ können wir dann als reihenfolgeabhängige Ausführungszeiten interpretieren. Den Depotknoten 0 können wir als virtuellen Auftrag auffassen, der am Anfang und am Ende jeder Auftragssequenz steht. Als Zielfunktion steht i.d.R. nicht die Minimierung der Kosten im Vordergrund, sondern die Minimierung der Fertigstellungszeit des letzten Auftrags (Minimierung des Makespan).

Balas [Bal85] untersuchte die Struktur von Scheduling-Polyedern für den Fall, daß lediglich untere Schranken r_i (release dates) für die Zeitvariablen t_i gegeben sind. In diesem Zusammenhang wurden von ihm auch neue Klassen von Ungleichungen formuliert. Da diese Ungleichungen auch für das ATSPTW verwendet werden können, werden wir eine Klasse davon, die Klasse der sogenannten 2-Job-Ungleichungen (englisch: two-job-cuts), hier vorstellen.

Um der Tatsache Rechnung zu tragen, daß für Scheduling Probleme oft nur reihenfolgeunabhängige Ausführungszeiten p_i, aber keine Kosten c_{ij} gegeben sind, werden wir die Kosten c_{ij} so weit wie möglich den Ausführungszeiten hinzurechnen. Dazu ermitteln wir den größten Wert a_i bzw. b_i, der von jeder Zeile bzw. Spalte der Kostenmatrix $C = (c_{ij})$ abgezogen werden kann, so daß alle Werte c_{ij} nichtnegativ bleiben. Wir setzen dann

$$\tilde{p}_i := p_i + a_i + b_i.$$

Für je zwei Knoten $i, j \in V$ mit $r_j < r_i + \tilde{p}_i$ und $r_i < r_j + \tilde{p}_j$ lassen sich nun die **2-Job-Ungleichungen** wie folgt formulieren:

$$(\tilde{p}_i + r_i - r_j)t_i + (\tilde{p}_j + r_j - r_i)t_j \geq \tilde{p}_i\tilde{p}_j + r_i\tilde{p}_j + r_j\tilde{p}_i.$$

Bei diesen 2-Job-Ungleichungen betrachten wir jedoch nur reihenfolgeunabhängige Ausführungszeiten $\tilde{p}_i$ für die Aufträge $i \in V$.

Auf Grund unserer Interpretation des ATSPTW als 1-Maschinen-Scheduling Problem mit reihenfolgeabhängigen Ausführungszeiten interessieren wir uns für eine Formulierung der 2-Job-Ungleichungen, in der reihenfolgeabhängige Ausführungszeiten berücksichtigt werden. In [Bal85] wurden derartige Ungleichungen wie folgt formuliert:

$$(\vartheta_{ij} + r_i - r_j)t_i + (\vartheta_{ji} + r_j - r_i)t_j \geq \vartheta_{ij}\vartheta_{ji} + r_i\vartheta_{ji} + r_j\vartheta_{ij}.$$

Hierbei sei vorausgesetzt, daß $\vartheta_{ij} \leq \vartheta_{ik} + \vartheta_{kj}\ \forall i \neq j \neq k \neq i$. Andernfalls ersetzen wir ϑ_{ij} durch ω_{ij}, wobei ω_{ij} wieder das Gewicht des kürzesten Weges von i nach j sei. Die Gültigkeit der 2-Job-Ungleichungen bzgl. P_2^{TW} sieht man wie folgt. Angenommen, der Knoten i wird vor dem Knoten j ausgeführt und $t_i = r_i$ bzw. $t_j = r_i + \vartheta_{ij}$, was den frühestmöglichen Startzeiten der Knoten i bzw. j entspricht. Ersetzen wir die Zeitvariablen durch diese Startzeiten und multiplizieren die linke

Seite der 2-Job-Ungleichung aus, so erhalten wir die rechte Seite der Ungleichung. In diesem Fall ist die 2-Job-Ungleichung also mit Gleichheit erfüllt. Andernfalls ($t_i > r_i$ bzw. $t_j > r_i + \vartheta_{ij}$) ist die linke Seite der Ungleichung größer als die rechte Seite.

Verletzte Ungleichungen dieser Art lassen sich durch Enumeration aller Knotenpaare $i, j \in V$, die $r_j < r_i + \tilde{p}_i$ und $r_i < r_j + \tilde{p}_j$ erfüllen, finden. Wir haben einen entsprechenden Separierungsalgorithmus für die 2-Job-Ungleichungen implementiert.

6 Der Branch&Cut-Algorithmus

Der B&C-Algorithmus wurde ursprünglich von Ascheuer [Asc95] zur Lösung des AHPPTW (siehe Abschnitt 4.2) in der Programmiersprache C implementiert. Da sich das AHPP auf einfache Weise in ein ATSP transformieren läßt (siehe Seite 23), kann dieser Algorithmus nach entsprechenden Veränderungen auch zur Lösung des ATSPTW verwendet werden. Der B&C-Algorithmus wurde so konzipiert, daß er alternativ mit der Modellierung 1 oder mit der Modellierung 2 aufgerufen werden kann. Von mir wurde die Modellierung 3 implementiert und die damit verbundenen Änderungen im B&C-Algorithmus vorgenommen..

Das Grundgerüst des B&C-Algorithmus basiert auf einer Entwicklung von Jünger, Reinelt und Thienel [JRT95]. Sie haben ein Branch&Cut-Framework ABACUS entwickelt und implementiert, was allerdings nur Modellierungen mit binären Variablen unterstützt. Die erforderlichen Änderungen und Ergänzungen bzgl. der Modellierung 2 (neben den binären Variablen x_{ij} für alle $(i,j) \in A$ sind Zeitvariablen t_i für alle $i \in V \cup \{0\}$ involviert) wurden von Ascheuer und bzgl. der Modellierung 3 (hier kommen zu den x_{ij}-Variablen noch Zeitvariablen t_{ij} für alle Bögen $(i,j) \in A$ hinzu) von mir vorgenommen.

Von Stefan Thienel [JRT95] wurde mir freundlicherweise ein Flußdigramm für das von ihm in Zusammenarbeit mit Jünger und Reinelt entwickelte Grundgerüst des B&C-Algorithmus zur Verfügung gestellt (siehe Abbildung 6.1). Es ist zu beachten, daß dort von Maximierungsproblemen ausgegangen wurde, so daß wir die Begriffe "obere Schranke" (upper bound) und "untere Schranke" (lower bound) zu vertauschen haben.

Eine Besonderheit des B&C-Algorithmus ist die Verwendung der Methode der dünnen Graphen (Sparse-Graph-Technik), die von Grötschel und Holland [GH85] eingeführt wurde. In der Abbildung 6.1 sind die entsprechenden Schritte des B&C-Algorithmus grau schraffiert. Wir wählen aus der Bogenmenge A des vollständigen Digraphens $D = (V \cup \{0\}, A)$ eine "vielversprechende" Teilmenge A' und setzen $x_{ij} = 0$ für alle $(i,j) \in A \backslash A'$ (nicht aktive Variablen). Die Variablen x_{ij} mit $(i,j) \in A'$ (aktive Variablen) nehmen wir in unser Anfangs-LP auf. Die Bogenmenge A' erhalten wir dadurch, indem wir für jeden Knoten die k bzgl. der Kosten c_{ij} kürzesten Bögen, die mit ihm inzident sind, auswählen (wir betrachten also jeweils nur die k nächsten Nachbarknoten). Den Digraphen $D' = (V \cup \{0\}, A')$ nennen wir **Sparse-Digraphen**. Er enthält wesentlich weniger Bögen ($|A'| = O(|V|)$) als der vollständige Digraph ($|A| = O(|V|^2)$). Wir halten damit die zu lösenden LP-Relaxierungen relativ klein. Damit der Sparse-Digraph auch hamiltonsch ist, ergänzen wir ihn um die Bögen eines zulässigen Hamiltonkreises, den wir mittels Anfangs- und Verbesserungsheuristiken erhalten. Um zu überprüfen, ob eine LP-Lösung bzgl. des Sparse-Digraphens auch bzgl. des vollständigen Digraphens optimal ist, müssen wir die nicht aktiven Va-

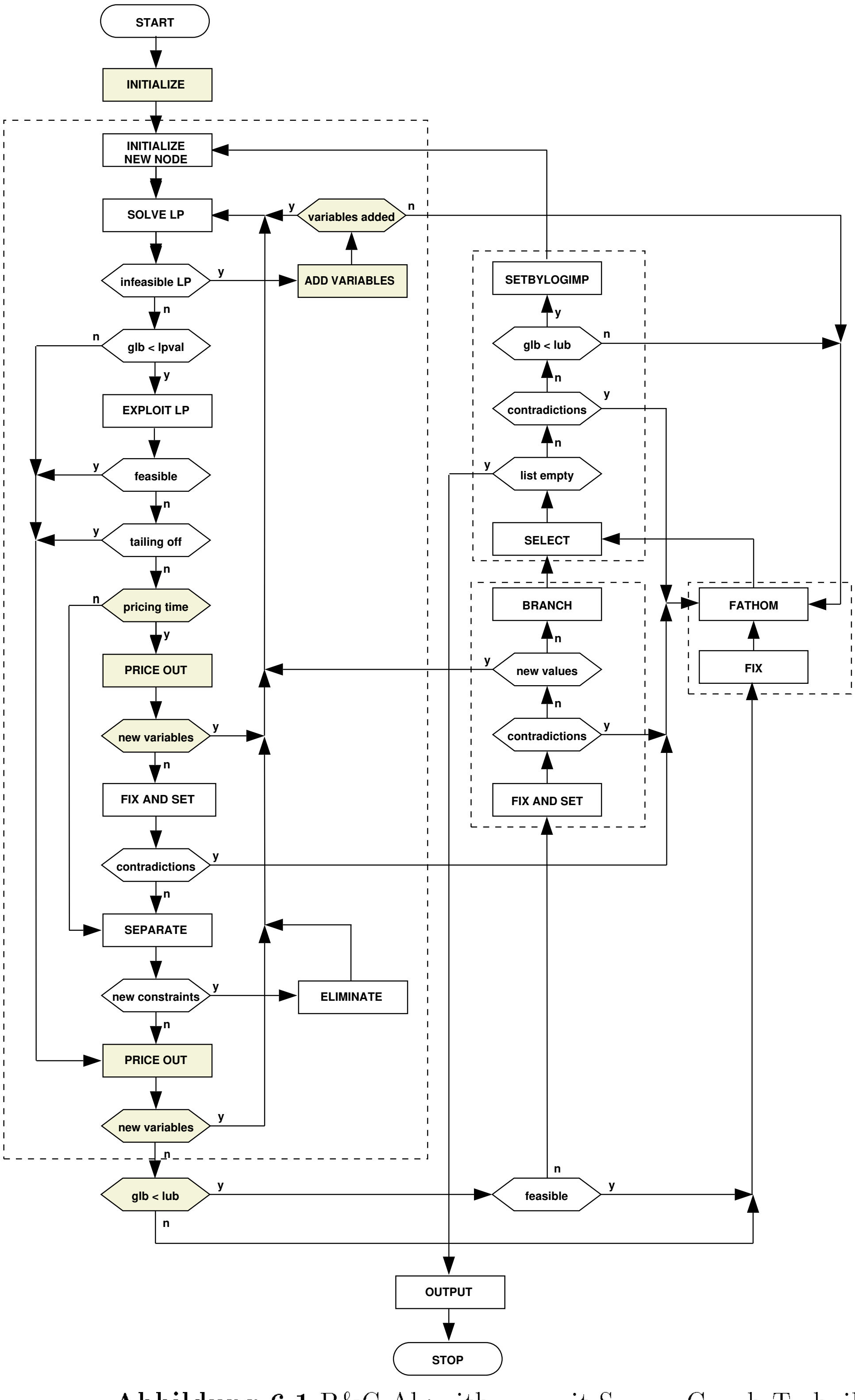

Abbildung 6.1 B&C-Algorithmus mit Sparse-Graph-Technik

riablen hinsichtlich ihrer reduzierten Kosten überprüfen (PRICE OUT). Sind alle reduzierten Kosten der nicht aktiven Variablen nichtnegativ, so ist die LP-Lösung auch für den vollständigen Digraphen gültig. Andernfalls fügen wir die Bögen, die den Variablen mit negativen reduzierten Kosten entsprechen (hier steht eine Verbesserung des Zielfunktionswertes in Aussicht), dem Sparse-Digraphen hinzu (ADD VARIABLES). Um nicht jedesmal alle nicht aktiven Variablen überprüfen zu müssen, beschränken wir uns bei der Überprüfung zunächst auf den sogenannten **Reserve-Digraphen** (hier betrachten wir jeweils die m nächsten Nachbarknoten mit $m > k$). Nach jedem p-ten gelösten LP führen wir eine zusätzliche Überprüfung der nicht aktiven Variablen durch, um sich "lohnende" Bögen früher in das LP zu holen und damit eine Laufzeitverbesserung zu erzielen. In unserem B&C-Algorithmus haben wir für den Sparse-Digraphen $k = 4$, für den Reserve-Digraphen $m = 7$ und für das zusätzliche Überprüfen der nicht aktiven Variablen $p = 7$ gewählt.

Eine weitere Besonderheit des B&C-Algorithmus ist die Verwendung der von Jünger et al. [JRT95] vorgeschlagenen Datenstruktur *pool* . Der *pool* ist am Anfang des Algorithmus leer. Ungleichungen, die während des Schnittebenenverfahrens generiert werden, fügen wir der LP-Relaxierung hinzu und speichern sie darüber hinaus im *pool* ab. Eine Ungleichung $a^T x \leq a_0$, deren Slack $s := a_0 - a^T x$ eine bestimmte Größe (z.B. $s \geq \epsilon = 0.0001$) überschreitet, wird aus der LP-Relaxierung entfernt (ELIMINATE), um deren Größe hinsichtlich der Anzahl der Nebenbedingungen moderat zu halten. Die Ungleichung bleibt jedoch im *pool* erhalten. Wir nennen eine Ungleichung aktiv, falls sie sowohl im *pool* als auch in der LP-Relaxierung steht. Eine Ungleichung, die nur im *pool* steht, heißt inaktiv. Inaktive Ungleichungen, die in einem der folgenden Iterationsschritte wiederum von der LP-Lösung verletzt werden, können nun aus dem *pool* generiert werden, ohne daß wir den Separierungsalgorithmus nochmals aufzurufen brauchen.

In den folgenden Abschnitten gehen wir kurz auf die drei Hauptbestandteile des B&C-Algorithmus enumerativer Rahmen, Berechnung der unteren Schranken und Berechnung der oberen Schranken, in die sich der Algorithmus zerlegen läßt, ein.

6.1 Enumerative Rahmen

Die wesentlichen Bestandteile des enumerativen Rahmens sind

- INITIALIZE, BRANCH, SELECT, FATHOM,
  FIX AND SET und OUTPUT.

In INITIALIZE berechnen wir mittels Anfangs- und Verbesserungsheuristiken einen zulässigen Hamiltonkreis (dessen Gewicht ist die anfängliche globale obere Schranke). Desweiteren konstruieren wir den Sparse- und Reserve-Digraphen.

In BRANCH wählen wir die Branching-Variable aus und generieren zwei neue Branch&Cut-Knoten, welche die Söhne des aktuellen Branch&Cut-Knotens werden. In dem ersten Sohn setzen wir die Branching-Variable auf 1, in dem zweiten Sohn auf 0. In allen drei Modellierungen wählen wir als Branching-Variable diejenige x_{ij}-Variable aus, die dem Wert 0.5 am nächsten liegt. Sind dies mehrere Variablen, so wählen wir diejenige mit dem höchsten Kostenkoeffizienten. In [JRT95] sind weitere Strategien zur Auswahl der Branching-Variablen zu finden.

In SELECT wählen wir den als nächstes zu bearbeitenden aktiven Branch&Cut-Knoten aus. Ist die Liste der aktiven Branch&Cut-Knoten leer, so ist die beste gefundene zulässige Lösung die Optimallösung, und wir gehen zu OUTPUT. Zur Auswahl des als nächstes zu bearbeitenden aktiven Branch&Cut-Knotens gibt es verschiedene Enumerations-Strategien. Wir verwenden in unserem B&C-Algorithmus die Methode Depth-First-Search, d.h., wir gehen in unserem Branch-&Cut-Baum so weit wie möglich in die Tiefe.

In FATHOM werden Branch&Cut-Knoten von den weiteren Betrachtungen ausgeschlossen. Ein Branch&Cut-Knoten braucht nicht mehr betrachtet zu werden, falls eine der folgenden Bedingungen zutrifft:

- die lokale untere Schranke ist größer oder gleich der globalen oberen Schranke (llb $\geq$ gub),
- ein unzulässiger Branch&Cut-Knoten wurde generiert, d.h., dieses Teilproblem enthält keine zulässige Lösung,
- die LP-Lösung ist Inzidenzvektor eines zulässigen Hamiltonkreises.

In FIX AND SET werden bestimmte x_{ij}-Variablen zu 0 bzw. 1 fixiert oder gesetzt. Das Fixieren ist für den gesamten weiteren Verlauf des Algorithmus gültig, das Setzen lediglich für den an dem aktuellen Branch&Cut-Knoten hängenden Teilbaum. Die Kriterien für das Fixieren bzw. Setzen sowie weitere Details sind in [JRT95] zu finden.

In OUTPUT geben wir die Optimallösung, d.h. die beste gefundene zulässige Lösung, aus und beenden den Algorithmus.

6.2 Berechnung der unteren Schranken

Der entscheidende Teil des B&C-Algorithmus ist die Berechnung der unteren Schranken mittels des Schnittebenenverfahrens (siehe Abschnitt 3.1). Die wesentlichen Schritte des Schnittebenenverfahrens sind entsprechend Abbildung 6.1 SOLVE LP und SEPARATE . Folgende strategische Überlegungen sollten bei der Implementierung des Schnittebenenverfahrens angestellt werden:

a) Kann durch Preprocessing die Anzahl der Variablen reduziert werden?

b) Welche Gleichungen und Ungleichungen sollen im Anfangs-LP enthalten sein?

c) Welche Klassen von Ungleichungen kommen für das Schnittebenenverfahren in Frage?

d) In welcher Reihenfolge sollen die Separierungsalgorithmen für die einzelnen Klassen von Ungleichungen aufgerufen werden?

6.2.1 Preprocessing

Das Preprocessing ist eine nicht zu unterschätzende Vorarbeit zur Erhöhung der Effizienz eines Lösungsalgorithmus für ein gegebenes kombinatorisches Optimierungsproblem $(E, \mathcal{I}, c)$. Auf Grund der gegebenen Daten eines Problembeispiels kann man oft bestimmte Elemente der Grundmenge E von den weiteren Betrachtungen ausschließen oder auch bestimmte Nebenbedingungen verschärfen. Von Ascheuer [Asc95] wurden verschiedene Möglichkeiten des Preprocessing für das AHPPTW untersucht und implementiert, die in unserem B&C-Algorithmus zur Lösung des ATSPTW ebenfalls benutzt werden können. Das Preprocessing wird wie folgt durchgeführt:

- Verkleinerung der gegebenen Zeitfenster,
- Ableitung von Reihenfolgebedingungen aus den gegebenen Zeitfenstern,
- Eliminierung von Bögen, die in keiner zulässigen Lösung enthalten sein können.

6.2.2 Anfangs-LP

Das Anfangs-LP für das Schnittebenenverfahren wird mit den Variablen, die mit den Bögen des Sparse-Digraphens korrespondieren, initialisiert. Für die Modellierung 2 fügen wir darüber hinaus Zeitvariablen t_i für jeden Knoten hinzu. Bei der Modellierung 3 kommt zu jeder Variablen x_{ij} eine Zeitvariable t_{ij} hinzu. Für alle drei Modellierungen stehen die Gradbeschränkungen

$$\begin{aligned} x(\delta^+(i)) &= 1 \quad \forall i \in V \cup \{0\} \\ x(\delta^-(i)) &= 1 \quad \forall i \in V \cup \{0\} \end{aligned} \tag{6.1}$$

im Anfangs-LP.

Bei der Modellierung 2 kommen die MTZ-Ungleichungen

$$t_i + \vartheta_{ij} - (1 - x_{ij})M \quad \leq \quad t_j \quad \forall (i,j) \in A,\ j \neq 0 \tag{6.2}$$

in das Anfangs-LP hinzu. Die Zeitfensterrestriktionen $r_i \leq t_i \leq d_i$ werden als untere bzw. obere Schranke für die Zeitvariablen t_i direkt an den LP-Löser übergeben. Zur Untersuchung der verschärften MTZ-Ungleichungen (siehe Unterabschnitt 5.6.1) nehmen wir diese anstelle der Ungleichungen (6.2) in das Anfangs-LP mit auf. Wenn wir die gelifteten Zeitfensterrestriktionen (siehe Unterabschnitt 5.6.2) untersuchen, so kommen diese zusätzlich in das Anfangs-LP hinzu.

Bei der Modellierung 3 haben wir neben den Gradbeschränkungen die Ungleichungen (4.9) (3) und (4), d.h.

$$\begin{aligned} \sum_{\substack{i=1\\ i\neq j}}^{n} t_{ij} + \sum_{\substack{i=0\\ i\neq j}}^{n} \vartheta_{ij} x_{ij} \quad &\leq \quad \sum_{\substack{k=0\\ k\neq j}}^{n} t_{jk} \quad j = 1, \ldots, n \quad \text{und} \\ r_i x_{ij} \quad \leq \quad t_{ij} \quad &\leq \quad d_i x_{ij} \qquad i,j = 0, \ldots, n,\ i \neq j,\ i \neq 0 \end{aligned} \tag{6.3}$$

im Anfangs-LP zu stehen.

6.2.3 Separierungs-Strategie

In dem Kapitel 5 haben wir verschiedene Klassen von Ungleichungen für das ATSPTW kennengelernt. Für diese Klassen von Ungleichungen sind entsprechende Separierungsalgorithmen implementiert. Da viele der Separierungsalgorithmen einen festen Startknoten sowie einen festen Endknoten erfordern, wird der Depotknoten in die Knoten s (Startknoten) und d (Endknoten) aufgespalten. Die Zeitfenster dieser Knoten sind so gewählt, daß jede Tour in s beginnt und in d endet. Die Separierungsalgorithmen (einschließlich der Separierung aus dem *pool*) werden in der folgenden Reihenfolge aufgerufen:

(1) Kurzzyklus-Ungleichungen.

(2) *pool*-Separierung (siehe Seite 58).

(3) 2-Matching-Ungleichungen.

(4) π-Ungleichungen.

(5) (π, σ)-Ungleichungen.

(6) σ-Ungleichungen.

(7) Unzulässige-Wege-Beschränkungen.

(8) Spezial-Ungleichungen (siehe Seite 46).

(9) Geliftete 3-Kreis-Ungleichungen.

(10) LP-Ungleichungen (siehe unten).

Sobald von einem Separierungsalgorithmus Ungleichungen generiert wurden, fügen wir diese dem *pool* und der LP-Relaxierung hinzu. Die nachfolgenden Separierungsalgorithmen werden aus Laufzeitgründen dann nicht mehr aufgerufen.

Der Separierungsalgorithmus für die Unzulässige-Wege-Beschränkungen wird in [Asc95] beschrieben. Ebenfalls dort wird die Verwendung von sogenannten LP-Ungleichungen (englisch: LP-cuts) motiviert. Diese Ungleichungen sind gültig, falls der von der LP-Lösung x^* induzierte Unterdigraph (wir betrachten nur die Bögen (i,j) mit $x^*_{ij} > 0$) keine zulässige Lösung enthält. Die LP-Ungleichungen verwenden wir nur im ersten Branch&Cut-Knoten, d.h. in der Wurzel des Branch-&Cut-Baumes, hier jedoch in jeder Iteration des Schnittebenenverfahrens.

Neben den oben genannten Ungleichungen werden wir zusätzlich die verallgemeinerten π-Ungleichungen, die verschärften (π, σ)-Ungleichungen und die 2-Job-Ungleichungen hinsichtlich ihrer Auswirkungen auf unseren B&C-Algorithmus untersuchen.

6.2.4 Verlassen des Schnittebenenverfahrens

In dem Schnittebenenverfahren lösen wir iterativ LP-Relaxierungen und fügen in jedem Iterationsschritt neu generierte Schnittebenen der LP-Relaxierung hinzu, um eine bessere untere Schranke für den Optimalwert des jeweiligen Teilproblems zu erhalten. Das Schnittebenenverfahren wird beendet, falls eine der folgenden Bedingungen eintrifft:

- die lokale untere Schranke ist größer oder gleich der globalen oberen Schranke (llb $\geq$ gub),
- ein unzulässiger Branch&Cut-Knoten wurde generiert, d.h., dieses Teilproblem enthält keine zulässige Lösung,
- die LP-Lösung ist Inzidenzvektor eines zulässigen Hamiltonkreises,
- es werden keine verletzten Ungleichungen gefunden,
- die lokale untere Schranke vergrößert sich nur unwesentlich (in k Iterationen steigt der Zielfunktionswert um weniger als $p\%$ an), obwohl der LP-Relaxierung Schnittebenen hinzugefügt werden ("tailing off").

Bevor das Schnittebenenverfahren für einen Branch&Cut-Knoten (Teilproblem) beendet wird, überprüfen wir die nicht aktiven Variablen hinsichtlich ihrer reduzierten Kosten. Haben alle nicht aktiven Variablen reduzierte Kosten größer gleich Null, so gehen wir im Fall des Auftretens einer der ersten drei Bedingungen zu FATHOM, und der Branch&Cut-Knoten sowie der daran hängende Teilbaum gelten als ergründet. Die entsprechenden Teilprobleme brauchen nicht mehr betrachtet zu werden. Trifft eine der letzten beiden Bedingungen ein, so gehen wir zu BRANCH und generieren zwei neue Teilprobleme.

In unserem B&C-Algorithmus sind die Parameter für das "tailing off" so gewählt, daß wir das Schnittebenenverfahren verlassen und zu BRANCH gehen, falls während der letzten 10 Iterationen keine Verbesserung der unteren Schranke erzielt wurde oder falls während der letzten 20 Iterationen die Verbesserung der unteren Schranke weniger als 1% betrug.

6.3 Berechnung der oberen Schranken

6.3.1 Eröffnungs- und Verbesserungsheuristiken

Zur Berechnung der anfänglichen oberen Schranken verwenden wir **Eröffnungs- und Verbesserungsheuristiken**. Mit diesen primalen Heuristiken berechnen wir eine zulässige Lösung in INITIALIZE. Das Gewicht der zulässigen Lösung ist unsere anfängliche globale obere Schranke. Die Eröffnungs- und Verbesserungsheuristiken wurden von Ascheuer implementiert. Es ist dabei zu beachten, daß die Heuristiken ursprünglich für das AHPPTW verwendet worden sind und bei den von uns getesteten Datensätzen die Kosten der Bögen, die mit dem Depotknoten inzidieren, Null sind.

Wir verwenden die Heuristiken in unserem B&C-Algorithmus in folgender Reihenfolge :

- Eröffnungsheuristiken:
  (1) Sortiere die Knoten $i \in V$ in aufsteigender Reihenfolge $1, \dots, n$ und überprüfe, ob dies eine zulässige Lösung ergibt.
  (2) Sortiere die Knoten $i \in V$ in bzgl. r_i aufsteigender Reihenfolge und überprüfe, ob dies eine zulässige Lösung ergibt.
  (3) Sortiere die Knoten $i \in V$ in bzgl. d_i aufsteigender Reihenfolge und überprüfe, ob dies eine zulässige Lösung ergibt.
  (4) Sortiere die Knoten $i \in V$ in bzgl. $m_i := r_i + \frac{d_i - r_i}{2}$ aufsteigender Reihenfolge und überprüfe, ob dies eine zulässige Lösung ergibt.
  (5) Nearest-Neighbor-Heuristik.
  (6) Insertion-Heuristik.

- Verbesserungsheuristiken
  (7) Swap-Heuristik.
  (8) Or-Change-Heuristik.
  (9) 2-Node-Exchange-Heuristik.

Die ersten vier Eröffnungsheuristiken sind sehr einfache Heuristiken zum Überprüfen, ob bestimmte Sequenzen von Knoten eine zulässige Lösung ergeben. Die Nearest-Neighbor-Heuristik konstruiert einen zulässigen Hamiltonkreis wie folgt. Wir starten mit einem Knoten $i \in V$ und gehen zu dem bzgl. der Kosten ihm am nächsten liegenden Knoten. Von dort gehen wir wieder zu dem am nächsten liegenden Knoten usw., wobei die Zulässigkeit bzgl. der Zeitfenster berücksichtigt wird. Die so gefundene Sequenz von Knoten bildet zusammen mit dem Depotknoten einen zulässigen Hamiltonkreis. Für die Insertion-Heuristik sei der Depotknoten in einen Startknoten s und einen Endknoten d aufgespalten. Wir berechnen einen kürzesten Weg von s nach d und fügen Knoten, die noch nicht in dem Weg enthalten sind, nacheinander ein. Dabei wählen wir unter Berücksichtigung der Zulässigkeit bzgl. der Zeitfenster jeweils den Knoten, der den geringsten Kostenzuwachs zur Folge hat ("cheapest insertion"). Die beste zulässige Lösung, die wir aus den Anfangsheuristiken erhalten, nehmen wir als zulässige Anfangslösung für die Verbesserungsheuristiken.

Die Swap-Heuristik vertauscht benachbarte Knoten, solange sich die Kosten der zulässigen Lösung durch den Tausch verringern lassen. Die Or-Change-Heuristik ist ein Austauschverfahren nach Or (1976), bei dem ein Weg $(v_1, \ldots, v_k)$ mit fester Länge k, der in der zulässigen Lösung enthalten ist, zwischen zwei benachbarten Knoten i und j neu eingefügt wird. Die 2-Node-Exchange-Heuristik ist eine Verallgemeinerung der Swap-Heuristik, bei der wir nicht nur benachbarte Knoten austauschen. Die beste zulässige Lösung, die wir aus den Verbesserungsheuristiken erhalten, nehmen wir als zulässige Anfangslösung für unseren B&C-Algorithmus. Ihr Gewicht ist unsere anfängliche globale obere Schranke.

6.3.2 LP-Exploitation-Heuristik

Im Verlaufe des B&C-Algorithmus versuchen wir die globale obere Schranke zu verbessern. Dies geschieht in EXPLOIT LP (siehe Abbildung 6.1). In jeder Iteration des Schnittebenenverfahrens rufen wir eine **LP-Exploitation-Heuristik** auf, welche die LP-Lösung zur Konstruktion einer besseren zulässigen Lösung wie folgt ausnutzt. Es sei x^* eine gebrochene LP-Lösung. Wir konstruieren den von x^* induzierten Unterdigraphen $D^* = (V \cup \{0\}, A^*)$ mit $A^* = \{(i,j) \in A \mid x^*_{ij} > 0\}$ und enumerieren alle zulässigen Lösungen in D^*. Enthält D^* keine zulässige Lösung, so rufen wir die Nearest-Neighbor-Heuristik mit einer modifizierten Kostenmatrix $C' = (c'_{ij})$ mit $c'_{ij} = (1.0 - x^*_{ij})c_{ij}$ auf und konstruieren eine zulässige

Lösung. Hat diese Lösung bzgl. der Originalkosten ein Gewicht, das kleiner als die globale obere Schranke ist, so ersetzen wir die gegenwärtige zulässige Lösung durch die neue zulässige Lösung und verringern die globale obere Schranke auf die Kosten der neuen zulässigen Lösung.

In [Asc95] ist eine LP-Exploitation-Heuristik im Rahmen eines B&C-Algorithmus zur Lösung des SOP (Sequentiell Ordering Problem) zu finden. Dort werden zusätzlich zu der oben beschriebenen Herangehensweise Verbesserungsheuristiken zur Konstruktion einer neuen zulässigen Lösung mit geringeren Kosten verwendet. Die damit verbundene Verbesserung der globalen oberen Schranke kann zur Reduzierung des B&C-Baumes (bzgl. der Anzahl der Branch&Cut-Knoten) und somit zu Laufzeitverbesserungen führen. In [JRT95] sind weitere Ansätze für LP-Exploitation-Heuristiken angegeben.

7 Numerische Ergebnisse und Auswertung

Die Implementierung der Modellierung 3 und der Separierungsalgorithmen für die verallgemeinerten π-Ungleichungen, für die verschärften (π, σ)-Ungleichungen und für die 2-Job-Ungleichungen, die Hinzunahme der gelifteten Zeitfensterrestriktionen und der verschärften MTZ-Ungleichungen in das Anfangs-LP sowie die damit verbundenen Veränderungen in dem B&C-Algorithmus wurden von mir in der Programmiersprache C vorgenommen.

Die nachfolgend beschriebenen Datensätze für das ATSPTW wurden von mir auf einer SPARCstation 20 von Sun Microsystems mit 192 Megabyte Hauptspeicher getestet. Zur Lösung der LP-Relaxierungen dient uns der LP-Löser CPLEX, Version 4.0 (CPLEX Optimization Inc.).

7.1 Beschreibung der Datensätze

Wir untersuchen zwei verschiedene Typen von Datensätzen. Beim ersten Typ von Datensätzen sind die Zeitfenster für jeden Knoten aktiv. Im Gegensatz dazu lassen wir beim zweiten Typ von Datensätzen auch relaxierte Zeitfenster für eine bestimmte Anzahl von Knoten zu (siehe [Asc95]). Die Datensätze des ersten Typs (*rbg...*) kommen aus dem im Kapitel 4.2 beschriebenen Beispiel aus der Praxis (Minimierung der Leerfahrzeiten eines Regalbediengeräts). Die Datensätze mit zum Teil relaxierten Zeitfenstern sind zufällig generierte Datensätze, die auf dem Problembeispiel $br17.atsp$ aus der TSPLIB (siehe [Rei91]) bzw. auf dem Problembeispiel $rbg27a$ beruhen.

Für die Problembeispiele $br17a.m$, $1 \leq m \leq n = 17$, verwenden wir die Kostenmatrix von $br17.atsp$. Die Zahl m gibt an, wieviele Zeitfenster in dem jeweiligen Problembeispiel aktiv sind. Demnach entspricht $br17a.00$ einem ATSP-Problembeispiel und $br17a.17$ einem ATSPTW-Problembeispiel, bei dem alle Knoten aktive Zeitfenster haben. Für die Problembeispiele $br17a.m$, $1 \leq m \leq n$ sind die Weiten der Zeitfenster aus dem Intervall $[70, 140]$ und die Ausführungszeiten aus dem Intervall $[30, 80]$ per Zufall generiert worden. Für die Problembeispiele $rbg27.i.m$ mit $i \in \{a, b, c\}$ und $1 \leq m \leq n = 27$ verwenden wir die Kostenmatrix von $rbg27a$. Für die Problembeispiele $rbg27.a.m$ bzw. $rbg27.b.m$ bzw. $rbg27.c.m$ sind die Weiten der Zeitfenster aus dem Intervall $[10, 30]$ bzw. $[30, 50]$ bzw. $[50, 70]$ per Zufall generiert worden. Die Ausführungszeiten der Problembeispiele $rbg27.i.m$ sind aus dem Intervall $[15, 40]$ per Zufall generiert worden.

In [Asc95] ist die Auswirkung unterschiedlicher Zeitfensterweiten untersucht worden. Danach sind Problembeispiele mit relativ kleinen bzw. relativ großen Zeitfensterweiten hinsichtlich des Rechenaufwands leichter zu lösen als Problembeispiele mit mittleren Zeitfensterweiten ($rbg27.i.m$, $i \in \{a, b\}$). Die unterschiedliche

Anzahl von aktiven Zeitfenstern führt ebenfalls zu einem unterschiedlichen Laufzeitverhalten, wie es die nachfolgenden Rechenergebnisse zeigen. So kann man generell sagen, daß Problembeispiele mit sehr wenigen aktiven Zeitfenstern auf Grund ihrer "Nähe" zu einem ATSP-Problembeispiel hinsichtlich des Rechenaufwands leichter zu lösen sind. Gleiches gilt für Problembeispiele mit sehr vielen aktiven Zeitfenstern, da die Zeitfenster die Anzahl der zulässigen Lösungen erheblich reduzieren und ein sehr gutes Preprocessing zulassen.

7.2 Vergleich der Modellierungen

Wie zu erwarten war, führen die drei unterschiedlichen mathematischen Modellierungen für das ATSPTW zu einem unterschiedlichen Laufzeitverhalten des B&C-Algorithmus. Wir haben den B&C-Algorithmus alternativ mit den Modellierungen 1, 2 und 3 für die *rbg...*-Datensätze, für die $br17a.m$-Datensätze für $1 \leq m \leq n$ und für einige $rbg27.i.m$-Datensätze jeweils maximal 3 Stunden laufen lassen. Wurde innerhalb dieser Zeit keine Optimallösung gefunden, so wurde der B&C-Algorithmus abgebrochen (in den Tabellen mit $*$ gekennzeichnet). Die nachfolgende Tabelle zeigt einen Teil der erzielten Ergebnisse für die drei verschiedenen Modellierungen im direkten Vergleich.

	Modellierung 1			Modellierung 2			Modellierung 3		
Problem	#N	#LP	CPU	#N	#LP	CPU	#N	#LP	CPU
rbg16b	43	201	0:10.93	49	210	0:23.87	15	124	0:43.97
rbg19b	201	1280	1:26.63	129	979	1:46.35	17	199	0:47.05
rbg19c	133	509	0:28.60	107	413	0:53.83	15	97	0:56.75
rbg27a	21	193	0:24.52	19	192	0:44.32	13	124	3:01.72
rbg50a	5	93	0:53.18	31	523	93:52.23	7	89	5:47.02
rbg67a	9	114	1:24.50	?	?	—*	11	143	15:38.18
br17a.04	22669	60885	55:45.43	31156	78076	—*	193	828	3:35.53
br17a.05	19159	48596	56:51.20	27843	63193	167:24.13	23	101	0:33.03
br17a.06	18047	44426	50:53.48	34067	80776	174:30.27	49	209	0:55.85
br17a.07	14633	39502	37:50.43	22549	52196	108:51.65	31	118	0:34.98
br17a.08	6763	18525	17:03.85	7961	20343	40:02.40	23	126	0:38.75
br17a.09	135	422	0:23.23	199	623	1:03.63	15	76	0:22.58
rbg27.b.04	949	6301	17:17.10	823	5385	37:36.48	31	314	2:39.32
rbg27.b.08	1019	7884	21:05.13	1173	9370	53:32.78	25	314	3:13.93
rbg27.b.10	231	1839	6:05.88	155	1292	7:07.35	17	310	2:40.35
rbg27.c.13	487	4526	16:50.15	743	6145	48:14.00	39	432	6:43.87
rbg27.c.14	233	2120	5:03.13	283	2413	11:14.82	17	201	1:59.68
rbg27.c.18	1	50	0:06.63	1	50	0:14.27	1	52	0:41.92

#N: Anzahl der Branch&Cut-Knoten
#LP: Anzahl der gelösten LP-Relaxierungen
CPU: Rechenzeit in Minuten (CPU = Central Processing Unit)

? Siehe Seite 69.

Tabelle 7.1. Vergleich der Modellierungen

Die kompletten Ergebnisse der getesteten Datensätze sind im Anhang in den Tabellen 1 bis 9 zu finden. Bei der Analyse der numerischen Ergebnisse ist als erstes zu beobachten, daß sich die Modellierung 3 gegenüber den anderen beiden Modellierungen durch einen hinsichtlich der Anzahl der Branch&Cut-Knoten wesentlich kleineren Branch&Cut-Baum auszeichnet. Das Ergebnis ist offensichtlich eine Laufzeitersparnis, da wesentlich weniger Teilprobleme gelöst werden müssen. Damit verbunden ist auch die geringere Anzahl der gelösten LP-Relaxierungen. Besonders deutlich zeigt sich dieses Verhalten bei den Problembeispielen mit nur m aktiven Zeitfenstern, wobei $\frac{1}{4} \cdot n < m < \frac{3}{4} \cdot n$.

Als zweites wurde bei dem Testen der Datensätze deutlich, daß die auf den drei verschiedenen Modellierungen beruhenden LP-Relaxierungen deutliche Unterschiede im Schwierigkeitsgrad bzgl. ihrer Lösung aufweisen. So liegt der prozentuale Anteil der LP-Zeit (Rechenzeit zur Lösung der LP-Relaxierungen) an der Gesamtlaufzeit bei allen getesteten Datensätzen im Durchschnitt bei der Modellierung 1 bei 19 %, bei der Modellierung 2 bei 39 % und bei der Modellierung 3 bei 73%. Die nachfolgende Tabelle gibt den prozentualen Anteil der LP-Zeit an der Gesamtlaufzeit in Abhängigkeit von den getesteten Datensätzen für die drei Modellierungen an.

Modell	rbg...	br17a...	rbg27...
1	17%	23%	17%
2	37%	35%	45%
3	74%	70%	76%

Tabelle 7.2. Prozentualer Anteil der LP-Zeit

Es wurde festgestellt, daß bei der Modellierung 3 vor allem die Anwendung des primalen Simplexalgorithmus (nach dem Hinzufügen neuer Variablen zum LP) im Gegensatz zum dualen Simplexverfahren (nach dem Hinzufügen neuer Schnittebenen) erheblich Zeit gekostet hat. Die unterschiedlichen LP-Zeiten der drei verschiedenen Modellierungen erklären auch die Tatsache, daß bei den leichteren Datensätzen mit Laufzeiten weit unter einer Minute die Modellierung 1 den anderen beiden Modellierungen überlegen ist.

Ein Grund für diese unterschiedlichen Schwierigkeitsgrade beim Lösen der LP-Relaxierungen ist die unterschiedliche Anzahl der Variablen in den drei verschiedenen Modellierungen. Während wir in der Modellierung 1 mit jedem Bogen genau eine Variable assoziieren, kommen in der Modellierung 2 für jeden Knoten noch eine Zeitvariable hinzu. Bei der Modellierung 3 haben wir sogar doppelt so viele Variablen wie in der Modellierung 1. Ein weiterer Grund für die unterschiedlichen Schwierigkeitsgrade beim Lösen der LP-Relaxierungen ist die unterschiedliche Anzahl der Nebenbedingungen in den Anfangs-LP der drei verschiedenen

Modellierungen. Während für die Modellierung 1 lediglich die Gradbeschränkungen im Anfangs-LP stehen und die Kurzzyklus-Ungleichungen im Rahmen des Schnittebenenverfahrens generiert werden, kommt bei der Modellierung 2 für jeden Bogen eine MTZ-Ungleichung in das Anfangs-LP hinzu. Bei der Modellierung 3 steht zusätzlich zu den Gradbeschränkungen für jeden Knoten eine Ungleichung zur Vermeidung von Subtouren im Anfangs-LP. Darüber hinaus stehen die Zeitfensterrestriktionen (untere und obere Schranke für die Zeitvariablen) im Anfangs-LP.

Die soeben beschriebene unterschiedliche Anzahl der Spalten und Zeilen der Anfangs-LP der drei verschiedenen Modellierungen ist sicherlich ein wesentlicher Grund für die unterschiedlichen Schwierigkeitsgrade beim Lösen der LP-Relaxierungen. Darüberhinaus ergeben sich aber aus den drei verschiedenen Modellierungen auch unterschiedliche Probleme numerischer Art. So sind die numerischen Probleme bei der Modellierung 2 bzgl. des Wertes “big M” nicht zu beseitigen. Es ist zu beobachten, daß der Lösungsvektor x^* einer LP-Relaxierung für die Modellierung 2 tendenziell wesentlich mehr gebrochene Komponenten ($0 < x^*_{ij} < 1$) als die Lösungsvektoren der LP-Relaxierungen der anderen beiden Modellierungen hat. Aus diesem Grund hat der B&C-Algorithmus mit der Modellierung 2 bei größeren Datensätzen (50 Knoten und mehr) eine unverhältnismäßig lange Laufzeit. Hier nimmt die Berechnung der oberen Schranken (LP-Exploitation-Heuristik) den größten Teil der Rechenzeit ein. Insbesondere zeigt sich das bei dem Problembeispiel *rbg67a*. Der von einer LP-Lösung x^* induzierte Unterdigraph (wir betrachten nur die Bögen (i,j) mit $x^*_{ij} > 0$) ist hinsichtlich der Anzahl der Bögen derart groß, daß die Enumeration aller zulässigen Lösungen in dem induzierten Unterdigraphen schon in der Wurzel des B&C-Baumes das maximale Zeitlimit von 3 Stunden überschreitet und wir den Algorithmus per Hand abbrechen mußten. Dagegen zeichnet sich die Modellierung 1 insbesondere dadurch aus, daß die Komponenten der Lösungsvektoren der LP-Relaxierungen zum größten Teil Werte wie 0 oder 1 bzw. einige gebrochene Werte mit der Tendenz zu 1/4, 1/2 oder 3/4 annehmen.

Numerische Probleme wurden auch bei der Modellierung 3 in Zusammenhang mit den Datensätzen mit relaxierten Zeitfenstern festgestellt. Hier stellt sich die Frage nach einer vernünftigen rechnerinternen Darstellung des Zeitfensters $[0, \infty)$, da ∞ i.d.R. durch einen sehr großen Wert rechnerintern dargestellt wird. Durch die Wahl des rechnerinternen Wertes für ∞ in Abhängigkeit von den Datensätzen in einer Größenordnung, bei der die Information der relaxierten Zeitfenster nicht verloren geht, konnten diese numerischen Probleme beseitigt werden.

Zusammenfassend können wir sagen, daß der B&C-Algorithmus mit der Modellierung 1 für die Datensätze ohne relaxierte Zeitfenster (*rbg...*) die kürzesten Laufzeiten hat. Die Modellierungen 2 und 3 liefern bezüglich dieser Datensätze schlechtere Laufzeiten, wobei die Modellierung 3 der Modellierung 2 vorzuziehen ist, falls wir Problembeispiele mit einer Knotenanzahl von 50 und mehr Knoten lösen wollen (siehe $rbg50a$ und $rbg67a$). Bei den Datensätzen mit zum Teil relaxierten Zeitfenstern liefert der B&C-Algorithmus mit der Modellierung 3 überraschend die besten Laufzeitergebnisse. Hier erweist sich die enge Verknüpfung der x_{ij}-Variablen mit den Zeitvariablen t_{ij} als großer Vorteil gegenüber den anderen Modellierungen.

7.3 Einfluß der Schnittebenen

Der entscheidende Teil des B&C-Algorithmus ist die Berechnung der unteren Schranken mittels des Schnittebenenverfahrens. Eine große Bedeutung kommt hierbei den in dem Schnittebenenverfahren verwendeten Klassen von Ungleichungen zu. In dem Kapitel 5 haben wir verschiedene Klassen von Ungleichungen, die wir für das ATSPTW verwenden können, beschrieben.

Die Tabellen 10 bis 18 im Anhang geben die Anzahl der verschiedenen Ungleichungen an, die während des B&C-Algorithmus für die unterschiedlichen Datensätze in Abhängigkeit von der jeweiligen Modellierung generiert worden sind. Hinsichtlich ihrer Anzahl und ihres Einflusses auf den Zielfunktionswert der LP-Relaxierungen spielen die Unzulässige-Wege-Beschränkungen, die Kurzzyklus-Ungleichungen, die π-Ungleichungen und die (π, σ)-Ungleichungen im Schnittebenenverfahren die größte Rolle. Das Verhältnis der Anzahl der generierten Schnittebenen zueinander ist bzgl. aller drei Modellierungen in etwa gleich. Die nachfolgende Tabelle gibt die Anzahl der Schnittebenen für die verschiedenen Datensätze an, die während des B&C-Algorithmus mit der Modellierung 2 generiert worden sind.

Datensatz	SEC	TMC	PI	SIG	PISI	UWB	SPEZ	3KR
rbg...	788	16	910	217	1899	1240	135	7
$br17a$...	88749	6711	5538	63	888	398442	72	8180
$rbg27$...	7197	49	11976	1321	12026	7105	652	22
gesamt	96734	6776	18424	1601	14813	406787	859	8209

Tabelle 7.3. Anzahl der generierten Schnittebenen

Die Verwendung neuer Klassen von Ungleichungen in dem Schnittebenenverfahren und auch die Verschärfung bestimmter Ungleichungen können zu einer Verbesserung der unteren Schranken und somit zu einer Laufzeitverkürzung des B&C-Algorithmus führen.

In dem Kapitel 5 haben wir die Klasse der verallgemeinerten π-Ungleichungen, die Klasse der bzgl. der Zeitfenster verschärften (π, σ)-Ungleichungen und die Klasse der 2-Job-Ungleichungen vorgestellt. Für diese Klassen von Ungleichungen haben wir die Separierungsalgorithmen implementiert und hinsichtlich ihrer

Auswirkung auf den B&C-Algorithmus getestet. Desweiteren haben wir Testläufe für die verschärften MTZ-Ungleichungen und für die gelifteten Zeitfensterrestriktionen vorgenommen. Die Testläufe wurden mit den rbg...-Datensätzen und den $br17a$...-Datensätzen für die Modellierung 2 durchgeführt.

Verallgemeinerte π-Ungleichungen

Im Unterabschnitt 5.4.2 haben wir die Klasse der verallgemeinerten π-Ungleichungen eingeführt. Um zu garantieren, daß der Präzedenzdigraph nach dem Schrumpfen der 1-Wege azyklisch ist, rufen wir vor dem Separierungsalgorithmus für die verallgemeinerten π-Ungleichungen den Shrinking-Separierungsalgorithmus für die PCB-Ungleichungen auf. Wir verwenden die PCB-Ungleichungen anstelle der Spezial-Ungleichungen, so daß die Spezial-Ungleichungen nicht mehr in dem Schnittebenenverfahren vorkommen. Die Separierungsreihenfolge mit den verallgemeinerten π-Ungleichungen sieht nun wie folgt aus:

(1) Kurzzyklus-Ungleichungen.

(2) *pool*-Separierung.

(3) 2-Matching-Ungleichungen.

(4) π-Ungleichungen.

(5) PCB-Ungleichungen (siehe Seite 45).

(6) Verallgemeinerte π-Ungleichungen.

(7) (π, σ)-Ungleichungen.

(8) σ-Ungleichungen.

(9) Unzulässige-Wege-Beschränkungen.

(10) Geliftete 3-Kreis-Ungleichungen.

(11) LP-Ungleichungen.

Wir rufen den Separierungsalgorithmus für die verallgemeinerten π-Ungleichungen nur dann auf, falls zuvor keine verletzten Kurzzyklus-Ungleichungen, keine verletzten π-Ungleichungen und keine verletzten PCB-Ungleichungen gefunden worden sind.

Da die verallgemeinerten π-Ungleichungen auf den Reihenfolgebedingungen beruhen, die wir aus den gegebenen Zeitfenstern abgeleitet haben und somit eine Relaxierung der Zeitfenster darstellen, waren den Erwartungen an die verallgemeinerten π-Ungleichungen bzgl. einer Laufzeitverbesserung des B&C-Algorithmus

Grenzen gesetzt. Insofern wurden die Erwartungen durch die erzielten Laufzeitergebnisse des B&C-Algorithmus bei Verwendung der verallgemeinerten π-Ungleichungen übertroffen. Die nachfolgende Tabelle zeigt einige der Laufzeitergebnisse im Vergleich zu den Laufzeitergebnissen des B&C-Algorithmus ohne Verwendung der verallgemeinerten π-Ungleichungen. Die vollständigen Ergebnisse der getesteten Datensätze befinden sich in den Tabellen 19 und 20 im Anhang.

	ohne verallg. π-Ungleichungen			mit verallg. π-Ungleichungen				
Problem	#N	#LP	CPU	#N	#LP	#PCB	#gPI	CPU
rbg19b	129	979	1:46.35	1	40	12	20	0:04.43
rbg27a	19	192	0:44.32	23	184	6	93	0:44.33
rbg50a	31	523	93:52.23	3	66	3	51	10:16.95
br17a.05	27843	63193	167:24.13	21467	44820	0	65	122:59.73
br17a.06	34067	80776	174:30.27	17927	41816	0	120	103:31.02
br17a.07	22549	52196	108:51.65	16813	37239	1	97	76:22.00

#PCB: Anzahl der PCB-Ungleichungen
#gPI: Anzahl der verallgemeinerten π-Ungleichungen

Tabelle 7.4. Verallgemeinerte π-Ungleichungen

Die numerischen Ergebnisse zeigen, daß sich die Laufzeiten des B&C-Algorithmus durch die Verwendung der verallgemeinerten π-Ungleichungen zum Teil erheblich verringern, obwohl die Anzahl der generierten verallgemeinerten π-Ungleichungen im Vergleich zu den anderen Ungleichungen relativ klein ist. Bei den *rbg*...-Datensätzen erzielen wir in der Summe eine Laufzeitverbesserung von rund 55% (ohne *rbg*50*a*), obwohl die Zeit für die Generierung der verallgemeinerten π-Ungleichungen zusätzlich zu der Gesamtlaufzeit hinzukommt. Allein bei dem Problembeispiel *rbg*50*a* erzielen wir eine Laufzeitverbesserung von fast 90%. Bei den *br*17...-Datensätzen wirken sich die verallgemeinerten π-Ungleichungen besonders bei schwierigen Problembeispielen (*br*17.04, ... , *br*17*a*.08) vorteilhaft aus. Hier liegt die Laufzeitverbesserug in der Summe bei rund 22%. Die Wirkung der verallgemeinerten π-Ungleichungen wird bei den Datensätzen mit relaxierten Zeitfenstern allerdings abgeschwächt, da wir hier weniger Reihenfolgebedingungen aus den Zeitfenstern ableiten können.

Die geringe Anzahl der verletzten PCB-Ungleichungen (siehe auch Tabellen 19 und 20 im Anhang) zeigt, daß der Präzedenzdigraph nach dem Schrumpfen der 1-Wege nur in wenigen Fällen gerichtete Kreise enthielt.

Zusammenfassend kann man sagen, daß die Verwendung der verallgemeinerten π-Ungleichungen zu den gewünschten Laufzeitverbesserungen geführt haben. Es ist aber zu erwähnen, daß auch das Austauschen der Spezial-Ungleichungen durch die PCB-Ungleichung sowie die Veränderung der Separierungsreihenfolge einen Anteil an der Laufzeitverbesserung haben können.

Verschärfte (π, σ)-Ungleichung

Für die verschärften (π, σ)-Ungleichungen ist derzeit kein exakter polynomialer Separierungsalgorithmus bekannt. Jedoch haben wir einen exakten Separierungsalgorithmus für die vereinfachte Version dieser Ungleichungen implementiert (vgl. Unterabschnitt 5.4.3). Dieser Separierungsalgorithmus basiert auf einem exakten Separierungsalgorithmus für die schwachen (π, σ)-Ungleichungen (siehe [Asc95]). Es seien $X, Y \subset V$, so daß $i \prec j$ für jedes Paar $i \in X, j \in Y$. Falls die Menge der bzgl. der Zeitfenster unzulässigen Bögen $Q = Q_1 \cup Q_2$, mit

$$Q_1 := \{(i,k),\,(k,j) \in A \mid \exists i \in X, j \in Y \text{ so daß der Weg } (i,k,j) \text{ unzulässig ist}\},$$
$$Q_2 := \{(u,v) \in A \mid \exists i \in X, j \in Y \text{ so daß der Weg } (i,u,v,j) \text{ unzulässig ist}\},$$

leer ist, reduziert sich die verschärfte (π, σ)-Ungleichung zu einer schwachen (π, σ)-Ungleichung. Der Separierungsalgorithmus generiert also in dem Fall, daß $Q = \emptyset$, eine schwache (π, σ)-Ungleichung und in dem Fall, daß $Q \neq \emptyset$, eine verschärfte (π, σ)-Ungleichung.

Die nachfolgende Tabelle zeigt einige Beispiele für die Auswirkung der verschärften (π, σ)-Ungleichungen. Die anderen Ergebnisse der getesteten Datensätze sind im Anhang in den Tabellen 21 und 22 aufgeführt.

	ohne verschärfte (π, σ)-Ungl.			mit verschärften (π, σ)-Ungl.			
Problem	#N	#LP	CPU	#N	#LP	#sPISI	CPU
rbg19c	107	413	0:53.83	93	373	5	0:48.40
rbg27a	19	192	0:44.32	1	27	4	0:06.65
rbg50a	31	523	93:52.23	31	523	0	94:14.72
br17a.06	34067	80776	174:30.27	32143	76585	16	178:54.65
br17a.07	22549	52196	108:51.65	8989	21281	5	45:34.78
br17a.08	7961	20343	40:02.40	5579	14249	13	28:56.97

#sPISI: Anzahl der verschärften (π, σ)-Ungleichungen

Tabelle 7.5. Verschärfte (π, σ)-Ungleichungen

Wir können anhand der getesteten Datensätze feststellen, daß die Verwendung der verschärften (π, σ)-Ungleichungen zum Teil zu einer erheblichen Verbesserung, aber auch zu einer Verschlechterung der Laufzeit des B&C-Algorithmus führen kann. So konnte das Problembeispiel *br17a.05* nicht mehr innerhalb des Zeitlimits von 3 Stunden gelöst werden. Bei den *rbg*...-Datensätzen wurden lediglich bei 2 Problembeispielen (*rbg19c*, *rbg27a*) verschärfte (π, σ)-Ungleichungen gefunden, die von einer LP-Lösung verletzt wurden.

Eine genauere Untersuchung hat ergeben, daß die Menge Q_1 für fast alle Problembeispiele leer war. Lediglich die Menge Q_2 ermöglichte eine Verschärfung der schwachen (π, σ)-Ungleichungen. Ganz offensichtlich hängt es stark von den

Weiten der gegebenen Zeitfenster ab, ob die Menge Q nicht leer ist und damit überhaupt verschärfte (π, σ)-Ungleichungen gefunden werden können. Bei den *rbg*...-Datensätzen haben wir im Vergleich zu der Summe aus Ausführungszeiten und Kosten sehr große Zeitfensterweiten gegeben. Das führt dazu, daß die kurzen Wege (i, k, j) und (i, u, v, j) so gut wie nie unzulässig werden. Diese Tatsache erklärt auch, warum wir bei den *rbg*...-Datensätzen so wenig verschärfte (π, σ)-Ungleichungen gefunden haben. Bei den *br17a*...-Datensätzen sind die Zeitfensterweiten dagegen aus dem Intervall $[70, 140]$ und somit wesentlich kleiner als die bei den *rbg*...-Datensätzen. Da bei den *rbg27*...-Datensätzen noch kleinere Zeitfensterweiten gegeben sind, liegt die Vermutung nahe, daß bei diesen Datensätzen größere Laufzeitverbesserungen zu erwarten sind. Die nachfolgende Tabelle (Legende siehe Anhang auf S. 81), die die Ergebnisse für die getesteten *rbg27*...-Datensätze beinhaltet, bestätigt voll und ganz diese Vermutung.

			B&C–Wurzel		B&C–Baum				
Problem	n	Opt.	Schranken	GAP	#N	Level	#LP	#sPISI	CPU
rbg27.a.06	29	161	–	–	1	0	32	1	0:08.40
rbg27.a.07	29	184	[182, 184]	0.01	3	1	61	2	0:15.98
rbg27.a.09	29	199	[198, 199]	0.01	3	1	130	8	0:47.57
rbg27.a.14	29	232	–	–	1	0	68	23	0:19.78
rbg27.a.17	29	265	–	–	1	0	33	14	0:05.05
rbg27.a.19	29	272	–	–	1	0	21	11	0:03.42
rbg27.b.04	29	170	[164, 176]	0.04	359	13	2569	283	20:59.10
rbg27.b.08	29	206	[193, 206]	0.07	605	16	4637	877	28:57.15
rbg27.b.10	29	210	[200, 212]	0.05	85	13	649	127	4:54.28
rbg27.b.13	29	224	[223, 224]	0.00	3	1	59	5	0:14.65
rbg27.b.15	29	234	–	–	1	0	53	23	0:15.55
rbg27.b.19	29	262	[261, 262]	0.00	7	3	77	24	0:15.23
rbg27.c.03	29	161	–	–	1	0	26	1	0:05.28
rbg27.c.04	29	162	[161, 177]	0.01	3	1	58	3	0:14.68
rbg27.c.13	29	249	[221, 249]	0.13	237	17	1789	525	16:37.20
rbg27.c.14	29	247	[219, 248]	0.13	71	12	405	115	2:56.25
rbg27.c.18	29	250	–	–	1	0	49	14	0:13.42
rbg27.c.19	29	250	–	–	1	0	27	14	0:06.72

Tabelle 7.6. Modellierung 2, verschärfte (π, σ)-Ungleichungen

Im Vergleich zu den Ergebnissen in der Tabelle 8 im Anhang haben sich die Laufzeiten des B&C-Algorithmus durch die Verwendung der verschärften (π, σ)-Ungleichungen erheblich verkürzt. In der Summe erhalten wir eine Laufzeitverbesserung von 54%, d.h., wir benötigen zur Lösung der *rgb27*...-Datensätze nur noch halb so viel Rechenzeit. Man vergleiche vor allem die Laufzeiten für die Problembeispiele *rbg27.b.04*, *rbg27.b.08* und *rbg27.c.13*.

Zusammenfassend können wir sagen, daß die verschärften (π, σ)-Ungleichungen für Datensätze mit (im Verhältnis zu der Summe aus Ausführungszeiten und Kosten) kleinen Zeitfensterweiten sehr gute Schnittebenen sind. Die getesteten

Datensätze haben ergeben, daß die verschärften (π, σ)-Ungleichungen dann zu einer erheblichen Verkürzung der Laufzeiten führen.

2-Job-Ungleichungen

Die 2-Job-Ungleichungen sind gültige Ungleichungen bzgl. des Polytops P_2^{TW}. Sie wurden im Unterabschnitt 5.6.3 vorgestellt. Wollen wir diese Klasse von Ungleichungen in unserem B&C-Algorithmus verwenden, so stellt sich die Frage nach ihrer Einordnung in die Reihenfolge der Separierung der bereits vorhandenen Klassen von Ungleichungen. Verschiedene Testläufe haben ergeben, daß die Einordnung der 2-Job-Ungleichungen nach den σ-Ungleichungen und vor den Unzulässige-Wege-Beschränkungen am sinnvollsten ist. Bei den Datensätzen mit zum Teil relaxierten Zeitfenstern werden die 2-Job-Ungleichungen hinsichtlich einer Verletzung nur dann untersucht, falls r_i und r_j für die entsprechenden Knoten i und j größer Null sind.

Die nachfolgende Tabelle gibt einen ersten Eindruck von den erzielten Ergebnissen. Die vollständigen Ergebnisse der getesteten Datensätze befinden sich im Anhang in den Tabellen 23 und 24.

	ohne 2-Job-Ungleichungen			mit 2-Job-Ungleichungen			
Problem	#N	#LP	CPU	#N	#LP	#2JOB	CPU
rbg19c	107	413	0:53.83	121	565	216	1:09.55
rbg27a	19	192	0:44.32	21	180	209	0:44.23
rbg50a	31	523	93:52.23	79	795	8	75:53.53
br17a.06	34067	80776	174:30.27	29635	85349	74444	177:12.77
br17a.07	22549	52196	108:51.65	20279	59515	50836	114:03.85
br17a.08	7961	20343	40:02.40	7581	25940	27470	52:12.87

#2JOB: Anzahl der 2-Job-Ungleichungen

Tabelle 7.7. 2-Job-Ungleichungen

Die Laufzeitergebnisse des B&C-Algorithmus bei Verwendung der 2-Job-Ungleichungen sind äußerst unbefriedigend. So konnte eine Laufzeitverbesserung nur in wenigen Fällen errreicht werden. Dagegen konnte das $br17a.05$-Problembeispiel nicht mehr innerhalb des Zeitlimits von 3 Stunden gelöst werden. Es stellte sich heraus, daß die 2-Job-Ungleichungen für die Datensätze mit zum Teil relaxierten Zeitfenstern ungeeignet sind. Da für viele Knoten $r_i = 0$ ist, werden keine 2-Job-Ungleichungen generiert (vgl. $br17a.04$). Durch das Preprocessing wird jedoch für viele Knoten mit relaxiertem Zeitfenster r_i auf einen Wert größer Null gesetzt (was bei dem Problembeispiel $br17a.04$ nicht der Fall war). Wurden während des Schnittebenenverfahrens dann verletzte 2-Job-Ungleichungen gefunden, so hatte dies so gut wie keine Auswirkung auf den Zielfunktionswert der darauffolgenden LP-Relaxierung. Da jedoch sehr viele 2-Job-Ungleichungen ohne erhebliche

Wirkung generiert wurden (vgl. $br17a.05$ bis $br17a.08$), verlangsamte sich die Gesamtlaufzeit in der Summe um über 5%.

Die schwache Wirkung der 2-Job-Ungleichungen ist damit zu erklären, daß diese Ungleichungen lediglich die Zeitvariablen beeinflussen. Die schwache Verknüpfung der Zeitvariablen mit den x_{ij}-Variablen durch die MTZ-Ungleichungen in der Modellierung 2 läßt so gut wie keinen Einfluß der 2-Job-Ungleichungen auf die x_{ij}-Variablen zu.

Geliftete Zeitfensterrestriktionen

Im Unterabschnitt 5.6.2 haben wir gezeigt, wie sich die Zeitfensterrestriktionen in der Modellierung 2 verschärfen lassen. Nehmen wir die gelifteten Zeitfensterrestriktionen anstelle der ursprünglichen Zeitfensterrestriktionen in das Anfangs-LP, so hoffen wir, eine bessere untere Schranke aus der entsprechenden LP-Relaxierung zu erhalten.

Die Rechenergebnisse in den Tabellen 25 und 26 im Anhang zeigen, daß die gewünschte Laufzeitverbesserung vor allem bei den leichten Problembeispielen mit Laufzeiten unter 2 Minuten ausgeblieben ist. Bei einigen Problembeispielen erhalten wir zwar eine größere untere Schranke in der Wurzel des Branch&Cut-Baumes (vgl. $rbg16b, rbg19c, rbg33a$), aber da sich das LP durch die Hinzunahme der gelifteten Zeitfensterrestriktionen vergrößert (für jeden Knoten $i \in V$ kommen 2 Ungleichungen hinzu), wird dafür mehr Zeit zur Lösung der LP benötigt. Somit verzeichnen wir in der Summe eine durchschnittliche Verschlechterung der Laufzeiten bei den leichten Problemen um bis zu 27%.

Bei den größeren Problembeispielen machen sich die gelifteten Zeitfensterrestriktionen positiv bemerkbar. Besonders hervorzuheben ist hierbei das Problembeispiel $rbg50a$. Hier erzielten wir eine Laufzeitverbesserung von über 95%. Die gelifteten Zeitfensterrestriktionen führten dazu, daß die Optimallösung schon im dritten Branch&Cut-Knoten auf dem Level 2 gefunden worden ist, während wir sonst einen sehr viel größeren am dritten Branch&Cut-Knoten hängenden Teilbaum (bis zum Level 9) zu untersuchen haben.

Verschärfte MTZ-Ungleichungen

Im Unterabschnitt 5.6.1 haben wir zwei verschiedene Möglichkeiten zur Verschärfung der Grundversion der MTZ-Ungleichungen kennengelernt. Zum einen kann die MTZ-Ungleichung für einen Bogen (i, j) bzgl. der Reihenfolgebedingung $i \prec j$ verschärft werden, und zum anderen kann sie bzgl. des Rückwärtsbogens (j, i) geliftet werden.

Um die verschärften MTZ-Ungleichungen zu testen, gehen wir bei der Generierung des Anfangs-LP für die Modellierung 2 wie folgt vor. Als erstes überprüfen wir, ob für die Knoten i und j eine Reihenfolgebedingung $i \prec j$ gegeben ist. Ist dies der Fall, so generieren wir die entsprechende verschärfte MTZ-Ungleichung. Andernfalls überprüfen wir, ob die Variable x_{ji} aktiv ist (d.h., ob der Rückwärtsbogen (j, i) im Sparse-Digraphen enthalten ist). Falls dies so ist, generieren wir die bzgl. des Rückwärtsbogens verschärfte MTZ-Ungleichung. Ist die Variable x_{ji} nicht aktiv, so greifen wir auf die Standardversion der MTZ-Ungleichung für den Bogen (i, j) zurück und schreiben diese in das Anfangs-LP. Es hat sich gezeigt, daß der Anteil der bzgl. der Rückwärtsbögen gelifteten MTZ-Ungleichungen den größten Anteil und die Grundversion der MTZ-Ungleichungen den kleinsten Anteil an den generierten MTZ-Ungleichungen haben. Die nachfolgende Tabelle beinhaltet den prozentualen Anteil der drei verschiedenen Versionen der MTZ-Ungleichungen in Abhängigkeit von den getesteten Datensätzen.

Datensatz	MTZ bzgl. $i \prec j$	MTZ bzgl. (j, i)	Grundversion
rbg...	29%	47%	24%
br17a...	24%	69%	7%

Tabelle 7.8. Prozentualer Anteil der verschiedenen MTZ-Ungleichungen

Die Tabellen 27 und 28 im Anhang enthalten die numerischen Ergebnisse bei der Verwendung der verschärften MTZ-Ungleichungen. Die Ergebnisse zeigen, daß sich die Laufzeiten des B&C-Algorithmus fast durchgängig verschlechtern. Besonders deutlich zeigt sich das am Problembeispiel *rbg50a* mit einer Laufzeitverschlechterung von über 77% und an den Problembeispielen *br17a.05* und *br17a.06*, die innerhalb des Zeitlimits von 3 Stunden nicht mehr gelöst werden konnten. Wir haben festgestellt, daß die Lösung der LP-Relaxierungen durch die verschärften MTZ-Ungleichungen schwieriger geworden ist. Dies schlägt sich in einer Verschlechterung der LP-Zeiten nieder. So stieg der prozentuale Anteil der LP-Zeiten an der Gesamtlaufzeit bei den *rbg...*-Datensätzen im Durchschnitt um 4% und bei den *br17a...*-Datensätzen im Durchschnitt um 10% an.

Abschließende Bemerkungen

Wie zu erwarten war, liefern die drei verschiedenen mathematischen Modellierungen für das ATSPTW zum Teil sehr unterschiedliche Laufzeitergebnisse. So hängt die Effizienz des B&C-Algorithmus stark von der verwendeten Modellierung ab. Deutliche Unterschiede zeigen sich bei der Lösung von Problembeispielen mit zum Teil relaxierten Zeitfenstern. Hier sind besonders die Ergebnisse der Modellierung 3 hervorzuheben. Sehr interessante Ergebnisse lieferten auch die neu implementierten Separierungsalgorithmen. Insbesondere konnte durch die Klasse der verallgemeinerten π-Ungleichungen sowie durch die Klasse der verschärften (π, σ)-Ungleichungen zum Teil erheblich die Gesamtlaufzeit des B&C-Algorithmus verringert werden. Die mathematische und algorithmische Analyse der drei verschiedenen Modellierungen für das ATSPTW sowie das Hinzunehmen weiterer Klassen von Ungleichungen zum Schnittebenenverfahren haben somit zu neuen Erkenntnissen geführt. Es liegt auf der Hand, daß noch größere Datensätze getestet werden müssen, um endgültige Aussagen zu treffen. Um aber einen Vergleich zu früheren Veröffentlichungen (vgl. [Asc95]) zu ermöglichen und nicht den Rahmen dieser Arbeit zu sprengen, haben wir uns auf die *rbg*...-, *br*17*a*...- und *rbg*27...-Datensätze beschränkt.

Es hat sich gezeigt, daß der B&C-Algorithmus ein effizientes, aber auch komplexes Instrument zur Lösung des ATSPTW ist. Die Analyse der polyedrischen Eigenschaften, insbesondere das Formulieren von gültigen Ungleichungen, ist ein wesentlicher Bestandteil der zu leistenden Vorarbeit. Aus der Verwendung des Schnittebenenverfahrens zur Berechnung der unteren Schranken ergeben sich strategische Fragestellungen, die es im Vorfeld zu klären gilt. Aber auch für die Berechnung der oberen Schranken sind gute Heuristiken zum Finden einer zulässigen Lösung (mit einem Gewicht möglichst nahe an dem Optimalwert) von großer Bedeutung.

Die bestehenden Schwierigkeiten bei der Lösung einiger Problembeispiele geben Anlaß zur weiteren Analyse verschiedener mathematischer Modellierungen des ATSPTW. So gibt es weiterhin einige Problembeispiele, die bis zur Fertigstellung der vorliegenden Arbeit mit keiner der drei Modellierungen optimal gelöst werden konnten. Die Untersuchung weiterer problemspezifischer Klassen von Ungleichungen bleibt unerläßlich, um bessere untere Schranken berechnen zu können. Es muß untersucht werden, ob durch einen Ausbau des Preprocessing die Größe der zu lösenden LP-Relaxierungen weiter reduziert und Nebenbedingungen verschärft werden können.

Der Entwurf neuer Eröffnungs- und Verbesserungsheuristiken für das ATSPTW kann zu einer Verbesserung der oberen Schranken und somit zu einer Verringerung der Laufzeit des B&C-Algorithmus führen. Eine Verbesserung der LP-Exploitation-Heuristik durch die Verwendung neuer Heuristiken für das ATSPTW kann ebenfalls zu einer Verbesserung der Laufzeit des B&C-Algorithmus führen. Hier hat sich die Enumeration aller zulässigen Lösungen in dem von der LP-Lösung induzierten Unterdigraphen bei Problembeispielen mit 50 und mehr Knoten für die Modellierung 2 als zu zeitaufwendig erwiesen.

Es bleibt zu untersuchen, ob durch die Verwendung einer anderen Enumerations-Strategie oder einer anderen Branching-Regel die Effizienz des B&C-Algorithmus zu erhöhen ist. Weiteren Spielraum bieten auch die anderen typischen Parameter des B&C-Algorithmus, die wir im Kapitel 6 beschrieben haben.

Abschließend möchte ich mich für die guten Arbeitsbedingungen, die mir im Konrad-Zuse-Zentrum für Informationstechnik Berlin hinsichtlich der Hard- und Softwareausstattungen, der Bibliotheksbetreuung und der flexiblen Arbeitszeiten zur Verfügung standen, bedanken. Doch was wären die besten Arbeitsbedingungen ohne einen Ansprechpartner in Situationen der Not. Daher gilt mein ganz besonderer Dank den wissenschaftlichen Mitarbeitern des Konrad-Zuse-Zentrums, allen voran Norbert Ascheuer, aber auch Ralf Borndörfer und Alexander Martin, die für fachliche Fragen immer ein offenes Ohr hatten. Für die intensive Korrekturlesung und die eine oder andere inhaltliche Anregung möchte ich mich bei meinem Mitstreiter und Studienkollegen Andreas Kokott bedanken. Last, not least möchte ich meinen Vater erwähnen, der dem Fehlerteufel in Sachen Grammatik und Rechtschreibung das Fürchten gelehrt hat.

Anhang

Legende zu den nachfolgenden Tabellen

Tabellen 1 bis 9

Problem:	Problembeispiel eines Datensatzes
n:	Anzahl der Knoten (inkl. Knoten s und d)
Opt.:	Wert der Optimallösung (opt)
Schranken:	$[lb, ub]$, untere und obere Schranke in der Wurzel des B&C-Baumes
GAP:	relative Abweichung der unteren Schranke lb von opt (GAP $= \frac{opt-lb}{lb}$)
#N:	Anzahl der Branch&Cut-Knoten
Level:	größte Level (gleich Höhe) des B&C-Baumes (Wurzel ist auf Level 0)
#LP:	Anzahl der gelösten LP-Relaxierungen
#Schnitte:	Gesamtanzahl der generierten Schnittebenen
CPU:	Rechenzeit in Minuten (CPU = Central Processing Unit)

Tabellen 10 bis 18

Problem:	Problembeispiel eines Datensatzes
POOL:	Anzahl der aus dem *pool* generierten Ungleichungen
SEC:	Anzahl der Kurzzyklus-Ungleichungen
TMC:	Anzahl der 2-Matching-Ungleichungen
PI:	Anzahl der π-Ungleichungen
SIG:	Anzahl der σ-Ungleichungen
PISI:	Anzahl der (π, σ)-Ungleichungen
UWB:	Anzahl der Unzulässige-Wege-Beschränkungen
SPEZ:	Anzahl der Spezial-Ungleichungen
3KR:	Anzahl der gelifteten 3-Kreis-Ungleichungen
LP:	Anzahl der LP-Ungleichungen (nur in der Wurzel)
Σ:	Gesamtanzahl der generierten Schnittebenen

Rechenergebnisse für die rbg...-Datensätze

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
rbg10a	12	149	–	–	1	0	9	19	0:00.15
rbg10b	12	108	–	–	1	0	1	0	0:00.08
rbg16a	18	179	–	–	1	0	8	14	0:00.37
rbg16b	18	142	[136, 142]	0.04	43	9	201	298	0:10.93
rbg17a	19	146	–	–	1	0	3	2	0:00.40
rbg19a	21	217	–	–	1	0	1	0	0:00.27
rbg19b	21	182	[175, 182]	0.04	201	13	1280	2162	1:26.63
rbg19c	21	190	[183, 190]	0.04	133	12	509	1461	0:28.60
rbg20a	22	210	–	–	1	0	1	0	0:00.87
rbg27a	29	268	[266, 268]	0.01	21	4	193	535	0:24.52
rbg33a	35	433	[433, 436]	0.00	3	1	40	119	0:06.53
rbg50a	52	414	[413, 433]	0.00	5	2	93	491	0:53.18
rbg67a	69	1048	[1047,1052]	0.00	9	4	114	531	1:24.50

Tabelle 1. Ergebnisse mit Modellierung 1

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
rbg10a	12	149	–	–	1	0	2	3	0:00.12
rbg10b	12	108	–	–	1	0	2	0	0:00.15
rbg16a	18	179	–	–	1	0	7	16	0:00.38
rbg16b	18	142	[135, 143]	0.05	49	8	210	311	0:23.87
rbg17a	19	146	–	–	1	0	4	8	0:00.62
rbg19a	21	217	–	–	1	0	1	0	0:00.33
rbg19b	21	182	[175, 182]	0.04	129	12	979	1616	1:46.35
rbg19c	21	190	[184, 190]	0.03	107	13	413	1139	0:53.83
rbg20a	22	210	–	–	1	0	1	0	0:00.95
rbg27a	29	268	[266, 268]	0.01	19	4	192	576	0:44.32
rbg33a	35	433	[421, 436]	0.03	3	1	55	173	0:18.40
rbg50a	52	414	[413, 433]	0.00	31	9	523	1578	93:52.23
rbg67a	69	?	[1043, 1062]	?	?	?	?	?	—*

Tabelle 2. Ergebnisse mit Modellierung 2

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
rbg10a	12	149	–	–	1	0	9	26	0:00.88
rbg10b	12	108	–	–	1	0	1	0	0:00.25
rbg16a	18	179	–	–	1	0	6	12	0:00.65
rbg16b	18	142	[136, 143]	0.04	15	7	124	211	0:43.97
rbg17a	19	146	–	–	1	0	3	0	0:02.50
rbg19a	21	217	–	–	1	0	1	0	0:00.53
rbg19b	21	182	[175, 182]	0.04	17	8	199	306	0:47.05
rbg19c	21	190	[184, 190]	0.03	15	5	97	211	0:56.75
rbg20a	22	210	–	–	1	0	1	0	0:01.35
rbg27a	29	268	[266, 268]	0.01	13	4	124	405	3:01.72
rbg33a	35	433	[421, 436]	0.03	3	1	31	103	0:26.58
rbg50a	52	414	[410, 433]	0.01	7	2	89	300	5:47.02
rbg67a	69	1048	[1042,1062]	0.01	11	5	143	573	15:38.18

Tabelle 3. Ergebnisse mit Modellierung 3

Rechenergebnisse für die br17a...-Datensätze

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
br17a.00	19	25	–	–	1	0	11	18	0:00.53
br17a.01	19	25	–	–	1	0	10	15	0:00.48
br17a.02	19	25	–	–	1	0	10	17	0:00.53
br17a.03	19	27	–	–	1	0	10	20	0:00.57
br17a.04	19	34	[27, 36]	0.26	22669	49	60885	97293	55:45.43
br17a.05	19	37	[30, 37]	0.23	19159	50	48596	92930	56:51.20
br17a.06	19	37	[30, 37]	0.23	18047	42	44426	87450	50:53.48
br17a.07	19	37	[30, 37]	0.23	14633	41	39502	77560	37:50.43
br17a.08	19	37	[30, 40]	0.23	6763	32	18525	35492	17:03.85
br17a.09	19	54	[49, 54]	0.10	135	12	422	903	0:23.23
br17a.10	19	55	[49, 55]	0.12	265	18	778	1389	0:43.43
br17a.11	19	55	[50, 55]	0.10	167	11	570	1306	0:30.17
br17a.12	19	57	[52, 57]	0.10	131	13	371	779	0:17.58
br17a.13	19	62	[54, 71]	0.15	67	7	218	691	0:09.42
br17a.14	19	62	[60, 62]	0.03	11	3	36	96	0:01.80
br17a.15	19	69	[60, 69]	0.15	17	5	63	185	0:02.52
br17a.16	19	81	–	–	1	0	1	0	0:00.45
br17a.17	19	81	–	–	1	0	1	0	0:00.23

Tabelle 4. Ergebnisse mit Modellierung 1

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
br17a.00	19	25	–	–	1	0	10	28	0:01.17
br17a.01	19	25	–	–	1	0	15	34	0:01.65
br17a.02	19	25	–	–	1	0	17	43	0:01.90
br17a.03	19	27	–	–	1	0	19	54	0:02.33
br17a.04	19	[27, 36]	[27, 36]	0.33	31156	54	78076	116473	—*
br17a.05	19	37	[30, 37]	0.23	27843	45	63193	105889	167:24.13
br17a.06	19	37	[30, 37]	0.23	34067	45	80776	156977	174:30.27
br17a.07	19	37	[30, 37]	0.23	22549	42	52196	91419	108:51.65
br17a.08	19	37	[30, 40]	0.23	7961	34	20343	38316	40:02.40
br17a.09	19	54	[49, 54]	0.10	199	12	623	1255	1:03.63
br17a.10	19	55	[49, 55]	0.12	303	20	941	1759	1:46.53
br17a.11	19	55	[50, 55]	0.10	223	14	771	1538	1:18.72
br17a.12	19	57	[52, 57]	0.10	63	11	255	587	0:19.93
br17a.13	19	62	[54, 71]	0.15	87	9	263	803	0:17.05
br17a.14	19	62	[60, 62]	0.03	7	2	31	149	0:02.00
br17a.15	19	69	[61, 69]	0.13	21	6	67	227	0:03.53
br17a.16	19	81	–	–	1	0	1	0	0:00.43
br17a.17	19	81	–	–	1	0	1	0	0:00.33

Tabelle 5. Ergebnisse mit Modellierung 2

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
br17a.00	19	25	–	–	1	0	16	37	0:06.52
br17a.01	19	25	–	–	1	0	8	18	0:02.25
br17a.02	19	25	–	–	1	0	13	39	0:02.53
br17a.03	19	27	–	–	1	0	41	103	0:11.08
br17a.04	19	34	[27, 36]	0.26	193	27	828	1204	3:35.53
br17a.05	19	37	[31, 37]	0.19	23	11	101	179	0:33.03
br17a.06	19	37	[31, 37]	0.19	49	8	209	373	0:55.85
br17a.07	19	37	[31, 37]	0.19	31	8	118	190	0:34.98
br17a.08	19	37	[31, 40]	0.19	23	10	126	216	0:38.75
br17a.09	19	54	[50, 54]	0.08	15	7	76	208	0:22.58
br17a.10	19	55	[50, 55]	0.10	31	9	135	338	0:29.88
br17a.11	19	55	[50, 55]	0.10	21	7	156	391	0:39.07
br17a.12	19	57	[52, 57]	0.10	13	6	75	198	0:13.33
br17a.13	19	62	[53, 71]	0.17	15	5	69	340	0:08.40
br17a.14	19	62	[58, 62]	0.07	5	2	14	46	0:01.78
br17a.15	19	69	[60, 69]	0.15	3	1	20	103	0:02.07
br17a.16	19	81	–	–	1	0	1	0	0:00.60
br17a.17	19	81	–	–	1	0	1	0	0:00.32

Tabelle 6. Ergebnisse mit Modellierung 3

Rechenergebnisse für die rbg27...-Datensätze

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
rbg27.a.06	29	161	–	–	1	0	30	67	0:03.27
rbg27.a.07	29	184	[182, 190]	0.01	73	9	496	753	1:17.73
rbg27.a.09	29	199	[196, 214]	0.02	7	2	149	345	0:18.98
rbg27.a.14	29	232	[231, 232]	0.00	3	1	119	309	0:41.27
rbg27.a.17	29	265	[262, 276]	0.01	13	5	109	313	0:08.93
rbg27.a.19	29	272	[271, 274]	0.00	3	1	29	110	0:02.85
rbg27.b.04	29	170	[164, 176]	0.04	949	19	6301	7654	17:17.10
rbg27.b.08	29	206	[194, 206]	0.06	1019	20	7884	10976	21:05.13
rbg27.b.10	29	210	[197, 212]	0.07	231	15	1839	2407	6:05.88
rbg27.b.13	29	224	[221, 224]	0.01	9	4	136	213	0:28.70
rbg27.b.15	29	234	[230, 241]	0.02	19	5	198	311	0:29.60
rbg27.b.19	29	262	[254, 262]	0.03	19	6	140	241	0:14.80
rbg27.c.03	29	161	–	–	1	0	36	65	0:03.62
rbg27.c.04	29	162	[161, 177]	0.01	3	1	72	146	0:08.13
rbg27.c.13	29	249	[213, 249]	0.17	487	17	4526	8741	16:50.15
rbg27.c.14	29	247	[218, 248]	0.13	233	12	2120	4529	5:03.13
rbg27.c.18	29	250	–	–	1	0	50	160	0:06.63
rbg27.c.19	29	250	–	–	1	0	34	110	0:04.40

Tabelle 7. Ergebnisse mit Modellierung 1

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
rbg27.a.06	29	161	–	–	1	0	33	85	0:08.98
rbg27.a.07	29	184	[182, 190]	0.01	107	11	650	972	4:11.57
rbg27.a.09	29	199	[195, 214]	0.02	5	2	130	271	0:26.37
rbg27.a.14	29	232	–	–	1	0	66	228	0:20.48
rbg27.a.17	29	265	[263, 270]	0.01	7	3	73	218	0:12.78
rbg27.a.19	29	272	[271, 279]	0.00	9	3	82	187	0:11.63
rbg27.b.04	29	170	[164, 176]	0.04	823	20	5385	6707	37:36.48
rbg27.b.08	29	206	[194, 206]	0.06	1173	20	9370	12414	53:32.78
rbg27.b.10	29	210	[200, 212]	0.05	155	14	1292	1705	7:07.35
rbg27.b.13	29	224	[222, 224]	0.01	9	4	169	264	1:44.12
rbg27.b.15	29	234	[232, 235]	0.01	13	5	170	281	1:41.10
rbg27.b.19	29	262	[259, 262]	0.01	19	6	152	255	0:33.05
rbg27.c.03	29	161	–	–	1	0	26	66	0:04.98
rbg27.c.04	29	162	[161, 177]	0.01	3	1	85	200	0:19.75
rbg27.c.13	29	249	[228, 249]	0.09	743	18	6145	11507	48:14.00
rbg27.c.14	29	247	[219, 248]	0.13	283	16	2413	5039	11:14.82
rbg27.c.18	29	250	–	–	1	0	50	148	0:14.27
rbg27.c.19	29	250	–	–	1	0	22	83	0:06.02

Tabelle 8. Ergebnisse mit Modellierung 2

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
rbg27.a.06	29	161	–	–	1	0	29	86	0:15.38
rbg27.a.07	29	184	[182, 190]	0.01	43	9	248	289	2:41.85
rbg27.a.09	29	199	[197, 199]	0.01	5	2	127	289	0:53.48
rbg27.a.14	29	232	[231, 232]	0.00	3	1	108	305	1:16.88
rbg27.a.17	29	265	[264, 265]	0.00	5	2	111	235	1:07.62
rbg27.a.19	29	272	[272, 274]	0.00	3	1	67	174	0:22.58
rbg27.b.04	29	170	[164, 176]	0.04	31	11	314	397	2:39.32
rbg27.b.08	29	206	[194, 206]	0.06	25	10	314	471	3:13.93
rbg27.b.10	29	210	[197, 212]	0.07	17	8	310	497	2:40.35
rbg27.b.13	29	224	[221, 224]	0.01	7	3	126	211	1:19.47
rbg27.b.15	29	234	[232, 235]	0.01	19	6	248	340	4:48.22
rbg27.b.19	29	262	[259, 262]	0.01	3	1	76	195	0:35.00
rbg27.c.03	29	161	–	–	1	0	38	101	0:13.20
rbg27.c.04	29	162	[161, 177]	0.01	3	1	63	173	0:33.45
rbg27.c.13	29	249	[229, 249]	0.09	39	12	432	877	6:43.87
rbg27.c.14	29	247	[229, 248]	0.08	17	6	201	420	1:59.68
rbg27.c.18	29	250	–	–	1	0	52	162	0:41.92
rbg27.c.19	29	250	–	–	1	0	24	89	0:22.65

Tabelle 9. Ergebnisse mit Modellierung 3

Anzahl der generierten Schnittebenen

Problem	POOL	SEC	TMC	PI	SIG	PISI	UWB	SPEZ	3KR	LP	Σ
rbg10a	0	10	0	0	0	0	8	0	0	1	19
rbg10b	0	0	0	0	0	0	0	0	0	0	0
rbg16a	0	4	0	0	0	0	10	0	0	0	14
rbg16b	68	59	0	42	4	66	98	0	0	29	298
rbg17a	0	2	0	0	0	0	0	0	0	0	2
rbg19a	0	0	0	0	0	0	0	0	0	0	0
rbg19b	598	166	7	513	266	664	272	254	0	20	2162
rbg19c	137	169	4	102	0	74	1090	0	2	20	1461
rbg20a	0	0	0	0	0	0	0	0	0	0	0
rbg27a	130	68	0	52	0	198	188	9	1	19	535
rbg33a	1	31	8	16	3	17	25	0	0	19	119
rbg50a	12	78	10	92	0	265	4	0	0	42	491
rbg67a	22	66	23	64	0	282	48	9	0	39	531

Tabelle 10. Schnittebenen mit Modellierung 1

Problem	POOL	SEC	TMC	PI	SIG	PISI	UWB	SPEZ	3KR	LP	Σ
rbg10a	0	2	0	0	0	0	0	0	0	1	3
rbg10b	0	0	0	0	0	0	0	0	0	0	0
rbg16a	0	5	0	0	0	0	6	0	0	5	16
rbg16b	90	55	0	65	1	87	78	0	0	25	311
rbg17a	0	6	0	0	0	0	0	0	0	2	8
rbg19a	0	0	0	0	0	0	0	0	0	0	0
rbg19b	615	129	3	429	217	479	199	120	1	39	1616
rbg19c	138	135	9	99	0	96	775	0	1	24	1139
rbg20a	0	0	0	0	0	0	0	0	0	0	0
rbg27a	180	65	2	66	0	299	112	9	3	20	576
rbg33a	26	55	2	30	0	42	22	0	0	22	173
rbg50a	552	336	0	221	0	896	48	15	2	60	1578
rbg67a	?	?	?	?	?	?	?	?	?	?	?

Tabelle 11. Schnittebenen mit Modellierung 2

Problem	POOL	SEC	TMC	PI	SIG	PISI	UWB	SPEZ	3KR	LP	Σ
rbg10a	0	12	0	1	0	0	6	0	0	7	26
rbg10b	0	0	0	0	0	0	0	0	0	0	0
rbg16a	0	4	0	0	0	0	6	0	0	2	12
rbg16b	78	23	0	34	5	55	50	0	1	43	211
rbg17a	0	0	0	0	0	0	0	0	0	0	0
rbg19a	0	0	0	0	0	0	0	0	0	0	0
rbg19b	156	47	0	86	15	61	42	36	0	19	306
rbg19c	51	37	0	22	0	29	98	0	0	25	211
rbg20a	0	0	0	0	0	0	0	0	0	0	0
rbg27a	55	34	0	24	0	170	136	0	1	40	405
rbg33a	0	48	2	9	1	5	18	0	0	20	103
rbg50a	14	150	0	53	0	71	6	0	0	20	300
rbg67a	90	276	0	85	0	172	0	0	0	40	573

Tabelle 12. Schnittebenen mit Modellierung 3

Problem	POOL	SEC	TMC	PI	SIG	PISI	UWB	SPEZ	3KR	LP	Σ
br17a.00	0	15	0	0	0	0	0	0	0	3	18
br17a.01	0	15	0	0	0	0	0	0	0	0	15
br17a.02	0	17	0	0	0	0	0	0	0	0	17
br17a.03	0	19	0	0	0	0	0	0	0	1	20
br17a.04	5172	13474	2463	262	15	168	79421	3	1467	20	97293
br17a.05	787	11180	1325	1011	1	28	78118	0	1247	20	92930
br17a.06	262	9739	1202	552	2	8	75026	7	894	20	87450
br17a.07	254	9223	1076	764	1	7	65646	0	823	20	77560
br17a.08	781	5858	532	884	1	128	27644	2	423	20	35492
br17a.09	32	150	10	91	0	39	567	7	20	19	903
br17a.10	135	229	14	144	3	136	805	0	38	20	1389
br17a.11	98	133	18	91	2	169	869	2	2	20	1306
br17a.12	51	85	10	60	0	67	533	0	1	23	779
br17a.13	7	56	6	23	0	24	552	0	3	27	691
br17a.14	0	5	2	0	0	4	74	0	0	11	96
br17a.15	2	14	2	1	0	5	142	0	0	21	185
br17a.16	0	0	0	0	0	0	0	0	0	0	0
br17a.17	0	0	0	0	0	0	0	0	0	0	0

Tabelle 13. Schnittebenen mit Modellierung 1

Problem	POOL	SEC	TMC	PI	SIG	PISI	UWB	SPEZ	3KR	LP	Σ
br17a.00	0	22	0	0	0	0	0	0	0	6	28
br17a.01	0	25	0	0	0	0	0	0	0	9	34
br17a.02	0	32	0	0	0	0	0	0	0	11	43
br17a.03	0	28	0	5	0	1	4	0	0	16	54
br17a.04	9156	21124	3683	99	19	138	89225	4	2141	40	116473
br17a.05	718	18214	1915	1509	13	25	81758	17	2418	20	105889
br17a.06	768	24923	2949	1146	9	15	125205	20	2690	20	156977
br17a.07	526	15287	1762	1192	8	0	71098	27	2025	20	91419
br17a.08	1368	8007	734	1165	3	94	27338	0	953	22	38316
br17a.09	57	234	18	116	4	56	758	3	48	18	1255
br17a.10	201	397	34	177	5	216	858	0	47	25	1759
br17a.11	201	224	22	63	1	240	941	0	20	27	1538
br17a.12	64	89	4	37	1	67	357	1	5	26	587
br17a.13	21	117	3	26	0	24	604	0	3	26	803
br17a.14	0	13	2	0	0	5	116	0	0	13	149
br17a.15	12	13	5	3	0	7	180	0	0	19	227
br17a.16	0	0	0	0	0	0	0	0	0	0	0
br17a.17	0	0	0	0	0	0	0	0	0	0	0

Tabelle 14. Schnittebenen mit Modellierung 2

Problem	POOL	SEC	TMC	PI	SIG	PISI	UWB	SPEZ	3KR	LP	Σ
br17a.00	0	29	0	0	0	0	0	0	0	8	37
br17a.01	0	14	0	0	0	0	0	0	0	4	18
br17a.02	0	28	0	0	0	0	0	0	0	11	39
br17a.03	6	44	1	4	0	1	18	0	0	35	103
br17a.04	152	348	19	25	8	16	626	5	113	44	1204
br17a.05	10	30	9	15	1	6	82	0	14	22	179
br17a.06	32	65	8	48	0	0	158	1	53	40	373
br17a.07	6	45	0	18	0	0	90	0	11	26	190
br17a.08	3	61	0	9	1	3	70	0	32	40	216
br17a.09	14	60	2	13	1	2	94	0	9	27	208
br17a.10	32	79	3	46	2	22	146	0	10	30	338
br17a.11	44	71	4	33	1	60	182	0	4	36	391
br17a.12	27	27	3	24	0	20	98	0	1	25	198
br17a.13	3	21	1	8	0	8	280	0	2	20	340
br17a.14	0	2	0	1	0	0	42	0	0	1	46
br17a.15	2	8	0	1	0	7	78	0	0	9	103
br17a.16	0	0	0	0	0	0	0	0	0	0	0
br17a.17	0	0	0	0	0	0	0	0	0	0	0

Tabelle 15. Schnittebenen mit Modellierung 3

Problem	POOL	SEC	TMC	PI	SIG	PISI	UWB	SPEZ	3KR	LP	Σ
rbg27.a.06	1	25	0	14	0	10	2	0	0	16	67
rbg27.a.07	135	194	2	268	29	102	118	0	0	40	753
rbg27.a.09	80	54	0	168	10	38	24	0	0	51	345
rbg27.a.14	183	39	2	138	4	81	6	0	0	39	309
rbg27.a.17	40	43	2	152	27	33	33	0	0	23	313
rbg27.a.19	0	20	0	50	0	9	18	0	0	13	110
rbg27.b.04	1925	1698	7	1662	622	1832	1737	53	23	20	7654
rbg27.b.08	3120	2131	25	2737	0	3991	2052	18	2	20	10976
rbg27.b.10	1267	282	3	698	0	992	408	4	0	20	2407
rbg27.b.13	108	31	0	72	0	61	6	0	0	43	213
rbg27.b.15	115	47	0	105	0	101	16	0	2	40	311
rbg27.b.19	180	14	0	122	0	58	2	4	0	41	241
rbg27.c.03	0	32	0	19	0	0	0	0	0	14	65
rbg27.c.04	16	44	0	48	1	13	20	0	0	20	146
rbg27.c.13	3192	735	6	3397	603	2804	1152	15	3	26	8741
rbg27.c.14	1043	403	7	1643	0	1181	1271	0	1	23	4529
rbg27.c.18	32	14	2	69	0	35	0	0	0	40	160
rbg27.c.19	34	14	1	50	0	18	0	0	0	27	110

Tabelle 16. Schnittebenen mit Modellierung 1

Problem	POOL	SEC	TMC	PI	SIG	PISI	UWB	SPEZ	3KR	LP	Σ
rbg27.a.06	4	36	0	14	0	10	0	0	0	25	85
rbg27.a.07	214	311	0	255	109	131	122	0	1	43	972
rbg27.a.09	89	49	0	112	5	40	20	0	0	45	271
rbg27.a.14	43	42	0	97	3	39	0	0	0	47	228
rbg27.a.17	87	20	0	100	1	26	34	0	0	37	218
rbg27.a.19	64	28	0	54	13	18	44	0	0	30	187
rbg27.b.04	1881	1934	12	1346	552	1482	1326	19	14	22	6707
rbg27.b.08	4771	2714	13	3010	0	4634	1993	8	2	40	12414
rbg27.b.10	762	208	0	485	0	638	303	8	0	63	1705
rbg27.b.13	203	31	0	75	0	71	4	0	0	83	264
rbg27.b.15	127	38	0	90	0	82	10	0	3	58	281
rbg27.b.19	135	22	0	113	0	57	4	6	0	53	255
rbg27.c.03	0	36	0	9	0	2	0	0	0	19	66
rbg27.c.04	16	85	0	34	0	7	14	0	0	60	200
rbg27.c.13	4935	1084	9	4340	638	3516	1745	17	2	156	11507
rbg27.c.14	1567	559	13	1738	0	1219	1486	0	0	24	5039
rbg27.c.18	53	18	1	49	0	43	0	0	0	37	148
rbg27.c.19	1	18	1	37	0	11	0	0	0	16	83

Tabelle 17. Schnittebenen mit Modellierung 2

Problem	POOL	SEC	TMC	PI	SIG	PISI	UWB	SPEZ	3KR	LP	Σ
rbg27.a.06	2	36	0	16	0	11	0	0	0	23	86
rbg27.a.07	43	115	0	50	20	24	38	0	0	42	289
rbg27.a.09	87	70	0	98	10	34	20	0	0	57	289
rbg27.a.14	131	43	0	126	6	77	0	0	0	53	305
rbg27.a.17	160	20	0	105	3	55	4	0	0	48	235
rbg27.a.19	50	16	0	70	8	21	18	0	0	41	174
rbg27.b.04	78	167	0	81	5	58	46	0	0	40	397
rbg27.b.08	130	211	0	103	0	68	27	0	0	62	471
rbg27.b.10	250	130	1	132	0	146	45	0	0	43	497
rbg27.b.13	116	25	0	71	0	64	6	0	0	45	211
rbg27.b.15	268	70	0	78	0	119	10	0	3	60	340
rbg27.b.19	111	20	0	86	0	30	2	0	0	57	195
rbg27.c.03	2	58	0	11	0	1	0	0	0	31	101
rbg27.c.04	3	99	0	20	0	6	8	0	0	40	173
rbg27.c.13	355	98	0	401	35	165	84	0	0	94	877
rbg27.c.14	186	64	2	132	0	81	93	0	0	48	420
rbg27.c.18	56	16	1	63	0	42	0	0	0	40	162
rbg27.c.19	3	12	1	45	0	14	0	0	0	17	89

Tabelle 18. Schnittebenen mit Modellierung 3

Verallgemeinerte π-Ungleichungen

			B&C–Wurzel		B&C–Baum					
Problem	n	Opt.	Schranken	GAP	#N	Level	#LP	#PCB	#gPI	CPU
rbg10a	12	149	–	–	1	0	2	0	0	0:00.05
rbg10b	12	108	–	–	1	0	2	0	0	0:00.12
rbg16a	18	179	–	–	1	0	7	0	0	0:00.48
rbg16b	18	142	[135, 143]	0.05	43	10	182	0	32	0:20.75
rbg17a	19	146	–	–	1	0	4	0	0	0:00.63
rbg19a	21	217	–	–	1	0	1	0	0	0:00.35
rbg19b	21	182	–	–	1	0	40	12	20	0:04.43
rbg19c	21	190	[184, 190]	0.03	41	9	190	0	20	0:23.40
rbg20a	22	210	–	–	1	0	1	0	0	0:00.98
rbg27a	29	268	[266, 268]	0.01	23	5	184	6	93	0:44.33
rbg33a	35	433	[421, 436]	0.03	3	1	47	0	21	0:18.28
rbg50a	52	414	[414, 433]	0.00	3	1	66	3	51	10:16.95

Tabelle 19. Modellierung 2, verallgemeinerte π-Ungleichungen

			B&C–Wurzel		B&C–Baum					
Problem	n	Opt.	Schranken	GAP	#N	Level	#LP	#PCB	#gPI	CPU
br17a.00	19	25	–	–	1	0	8	0	0	0:01.07
br17a.01	19	25	–	–	1	0	9	0	0	0:01.10
br17a.02	19	25	–	–	1	0	8	0	0	0:01.02
br17a.03	19	27	–	–	1	0	22	0	2	0:02.03
br17a.04	19	[28, 36]	[28, 36]	0.29	31971	44	71544	0	16	—*
br17a.05	19	37	[30, 37]	0.23	21467	49	44820	0	65	122:59.73
br17a.06	19	37	[30, 37]	0.23	17927	40	41816	0	120	103:31.02
br17a.07	19	37	[30, 37]	0.23	16813	39	37239	1	97	76:22.00
br17a.08	19	37	[30, 40]	0.23	6299	32	16129	1	95	42:19.02
br17a.09	19	54	[49, 54]	0.10	239	12	824	1	69	1:32.97
br17a.10	19	55	[49, 55]	0.12	319	17	987	1	116	1:55.23
br17a.11	19	55	[50, 55]	0.10	237	13	821	4	93	1:28.02
br17a.12	19	57	[52, 57]	0.10	77	13	253	1	24	0:20.20
br17a.13	19	62	[54, 71]	0.15	73	9	234	0	6	0:16.08
br17a.14	19	62	[60, 62]	0.03	7	2	32	0	0	0:02.27
br17a.15	19	69	[61, 69]	0.13	19	6	56	0	2	0:03.42
br17a.16	19	81	–	–	1	0	1	0	0	0:00.45
br17a.17	19	81	–	–	1	0	1	0	0	0:00.27

Tabelle 20. Modellierung 2, verallgemeinerte π-Ungleichungen

Legende:

Analog Tabellen 1 bis 9, bis auf
#PCB: Anzahl der PCB-Ungleichungen,
#gPI: Anzahl der verallgemeinerten π-Ungleichungen.

Verschärfte (π, σ)-Ungleichungen

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#sPISI	CPU
rbg10a	12	149	–	–	1	0	2	0	0:00.15
rbg10b	12	108	–	–	1	0	2	0	0:00.17
rbg16a	18	179	–	–	1	0	7	0	0:00.42
rbg16b	18	142	[135, 143]	0.05	49	8	210	0	0:24.25
rbg17a	19	146	–	–	1	0	4	0	0:00.60
rbg19a	21	217	–	–	1	0	1	0	0:00.27
rbg19b	21	182	[175, 182]	0.04	129	12	979	0	1:48.37
rbg19c	21	190	[184, 190]	0.03	93	11	373	5	0:48.40
rbg20a	22	210	–	–	1	0	1	0	0:01.02
rbg27a	29	268	–	–	1	0	27	4	0:06.65
rbg33a	35	433	[421, 436]	0.03	3	1	55	0	0:18.38
rbg50a	52	414	[413, 433]	0.00	31	9	523	0	94:14.72

Tabelle 21. Modellierung 2, verschärfte (π, σ)-Ungleichungen

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#sPISI	CPU
br17a.00	19	25	–	–	1	0	10	0	0:01.03
br17a.01	19	25	–	–	1	0	15	0	0:01.67
br17a.02	19	25	–	–	1	0	17	0	0:01.83
br17a.03	19	27	[25, 27]	0.08	3	1	25	1	0:03.47
br17a.04	19	[27, 34]	[27, 36]	0.26	26850	46	69104	11	—*
br17a.05	19	[30, 37]	[30, 37]	0.23	28615	47	66857	19	—*
br17a.06	19	37	[30, 37]	0.23	32143	43	76585	16	178:54.65
br17a.07	19	37	[30, 37]	0.23	8989	34	21281	5	45:34.78
br17a.08	19	37	[30, 40]	0.23	5579	29	14249	13	28:56.97
br17a.09	19	54	[49, 54]	0.10	113	12	399	39	0:45.23
br17a.10	19	55	[49, 55]	0.12	151	14	528	52	1:00.38
br17a.11	19	55	[50, 55]	0.10	121	14	379	46	0:47.45
br17a.12	19	57	[52, 57]	0.10	77	10	251	30	0:20.58
br17a.13	19	62	[55, 62]	0.13	35	7	123	21	0:08.58
br17a.14	19	62	[60, 62]	0.03	9	3	25	4	0:01.67
br17a.15	19	69	[63, 69]	0.10	11	5	42	7	0:02.68
br17a.16	19	81	–	–	1	0	1	0	0:00.42
br17a.17	19	81	–	–	1	0	1	0	0:00.28

Tabelle 22. Modellierung 2, verschärfte (π, σ)-Ungleichungen

Legende:
Analog Tabellen 1 bis 9, bis auf

#sPISI: Anzahl der verschärften (π, σ)-Ungleichungen.

2-Job-Ungleichungen

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#2JOB	CPU
rbg10a	12	149	–	–	1	0	2	0	0:00.13
rbg10b	12	108	–	–	1	0	2	0	0:00.08
rbg16a	18	179	–	–	1	0	7	0	0:00.58
rbg16b	18	142	[135, 143]	0.05	45	10	251	19	0:26.00
rbg17a	19	146	–	–	1	0	4	0	0:00.57
rbg19a	21	217	–	–	1	0	1	0	0:00.32
rbg19b	21	182	[175, 182]	0.04	143	11	882	27	1:45.70
rbg19c	21	190	[184, 190]	0.03	121	12	565	216	1:09.55
rbg20a	22	210	–	–	1	0	1	0	0:00.93
rbg27a	29	268	[266, 268]	0.01	21	5	180	209	0:44.23
rbg33a	35	433	[421, 436]	0.03	3	1	53	4	0:17.27
rbg50a	52	414	[413, 433]	0.00	79	12	795	8	75:53.53

Tabelle 23. Modellierung 2, 2-Job-Ungleichungen

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#2JOB	CPU
br17a.00	19	25	–	–	1	0	10	0	0:01.05
br17a.01	19	25	–	–	1	0	15	0	0:01.63
br17a.02	19	25	–	–	1	0	17	0	0:01.90
br17a.03	19	27	–	–	1	0	19	0	0:02.37
br17a.04	19	[27, 36]	[27, 36]	0.33	30796	53	75928	0	—*
br17a.05	19	[30, 37]	[30, 37]	0.23	25872	41	76198	65928	—*
br17a.06	19	37	[30, 37]	0.23	29635	41	85349	74444	177:12.77
br17a.07	19	37	[30, 37]	0.23	20279	40	59515	50836	114:03.85
br17a.08	19	37	[30, 40]	0.23	7581	32	25940	27470	52:12.87
br17a.09	19	54	[49, 54]	0.10	231	13	877	399	1:29.48
br17a.10	19	55	[49, 55]	0.12	303	19	1098	481	1:58.13
br17a.11	19	55	[49, 55]	0.12	223	13	884	228	1:27.20
br17a.12	19	57	[52, 57]	0.10	141	13	497	73	0:40.37
br17a.13	19	62	[54, 71]	0.15	83	8	295	65	0:18.82
br17a.14	19	62	[59, 62]	0.05	15	4	48	8	0:02.85
br17a.15	19	69	[62, 69]	0.11	31	8	91	5	0:04.37
br17a.16	19	81	–	–	1	0	1	0	0:00.45
br17a.17	19	81	–	–	1	0	1	0	0:00.33

Tabelle 24. Modellierung 2, 2-Job-Ungleichungen

Legende:
Analog Tabellen 1 bis 9, bis auf
#2JOB: Anzahl der 2-Job-Ungleichungen.

Geliftete Zeitfensterrestriktionen

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
rbg10a	12	149	–	–	1	0	7	23	0:00.47
rbg10b	12	108	–	–	1	0	1	0	0:00.15
rbg16a	18	179	–	–	1	0	7	13	0:00.67
rbg16b	18	142	[137, 143]	0.04	45	11	212	270	0:33.58
rbg17a	19	146	–	–	1	0	6	11	0:00.83
rbg19a	21	217	–	–	1	0	1	0	0:00.38
rbg19b	21	182	[175, 182]	0.04	95	10	821	1299	2:00.25
rbg19c	21	190	[185, 190]	0.03	111	12	483	1200	1:17.65
rbg20a	22	210	–	–	1	0	1	0	0:01.00
rbg27a	29	268	[266, 268]	0.01	25	5	199	612	0:54.82
rbg33a	35	433	[433, 436]	0.00	3	1	44	126	0:16.85
rbg50a	52	414	[410, 433]	0.01	5	2	82	293	3:46.12

Tabelle 25. Modellierung 2, geliftete Zeitfensterrestriktionen

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
br17a.00	19	25	–	–	1	0	11	30	0:01.90
br17a.01	19	25	–	–	1	0	15	34	0:03.17
br17a.02	19	25	–	–	1	0	12	35	0:02.17
br17a.03	19	27	[25, 27]	0.08	3	1	25	57	0:04.05
br17a.04	19	[27, 34]	[27, 36]	0.26	23212	35	63456	99308	—*
br17a.05	19	[30, 37]	[30, 37]	0.23	28212	48	67909	126952	—*
br17a.06	19	37	[30, 37]	0.23	26309	42	61558	114911	164:13.63
br17a.07	19	37	[30, 37]	0.23	20327	38	48132	83232	118:13.63
br17a.08	19	37	[30, 40]	0.23	4049	32	10714	21760	25:52.83
br17a.09	19	54	[49, 54]	0.10	161	13	604	1301	1:22.45
br17a.10	19	55	[49, 55]	0.12	255	20	763	1366	1:56.38
br17a.11	19	55	[50, 55]	0.10	219	13	753	1541	1:55.18
br17a.12	19	57	[52, 57]	0.10	143	14	475	1040	0:42.62
br17a.13	19	62	[54, 62]	0.15	79	8	261	944	0:18.52
br17a.14	19	62	[60, 62]	0.03	11	3	36	143	0:02.38
br17a.15	19	69	[60, 69]	0.15	21	6	68	244	0:03.57
br17a.16	19	81	–	–	1	0	1	0	0:00.42
br17a.17	19	81	–	–	1	0	1	0	0:00.28

Tabelle 26. Modellierung 2, geliftete Zeitfensterrestriktionen

Legende:
Analog Tabellen 1 bis 9.

Verschärfte MTZ-Ungleichungen

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
rbg10a	12	149	–	–	1	0	8	20	0:00.33
rbg10b	12	108	–	–	1	0	1	0	0:00.12
rbg16a	18	179	–	–	1	0	9	19	0:00.55
rbg16b	18	142	[136, 143]	0.04	53	9	220	336	0:25.97
rbg17a	19	146	–	–	1	0	3	3	0:00.47
rbg19a	21	217	–	–	1	0	1	0	0:00.35
rbg19b	21	182	[175, 182]	0.04	149	12	973	1565	1:58.12
rbg19c	21	190	[184, 190]	0.03	129	13	473	1307	1:08.20
rbg20a	22	210	–	–	1	0	1	0	0:00.98
rbg27a	29	268	[266, 268]	0.01	23	5	193	623	0:45.47
rbg33a	35	433	–	–	1	0	33	100	0:12.32
rbg50a	52	414	[411, 433]	0.01	117	9	1066	2802	166:40.47

Tabelle 27. Modellierung 2, verschärfte MTZ-Ungleichungen

Problem	n	Opt.	B&C–Wurzel Schranken	GAP	B&C–Baum #N	Level	#LP	#Schnitte	CPU
br17a.00	19	25	–	–	1	0	8	20	0:01.55
br17a.01	19	25	–	–	1	0	8	21	0:01.38
br17a.02	19	25	–	–	1	0	12	33	0:01.98
br17a.03	19	27	[25, 27]	0.08	3	1	26	58	0:04.32
br17a.04	19	[27, 34]	[27, 36]	0.26	23922	41	66512	107531	—*
br17a.05	19	[30, 37]	[30, 37]	0.23	25275	42	67729	135274	—*
br17a.06	19	[30, 37]	[30, 37]	0.23	28687	44	77504	177007	—*
br17a.07	19	37	[30, 37]	0.23	19355	41	50440	97602	116:35.67
br17a.08	19	37	[30, 40]	0.23	6627	33	17985	35320	40:42.98
br17a.09	19	54	[49, 54]	0.10	213	12	761	1750	1:44.73
br17a.10	19	55	[49, 55]	0.12	285	17	917	1861	1:59.55
br17a.11	19	55	[50, 55]	0.10	169	14	597	1300	1:13.05
br17a.12	19	57	[52, 57]	0.10	99	13	310	631	0:26.32
br17a.13	19	62	[54, 62]	0.15	75	8	199	544	0:15.03
br17a.14	19	62	[60, 62]	0.03	9	3	26	136	0:02.03
br17a.15	19	69	[63, 69]	0.10	19	6	62	214	0:03.88
br17a.16	19	81	–	–	1	0	1	0	0:00.50
br17a.17	19	81	–	–	1	0	1	0	0:00.33

Tabelle 28. Modellierung 2, verschärfte MTZ-Ungleichungen

Legende:
Analog Tabellen 1 bis 9.

Literatur

[AG88] A. Assad and B. Golden. *Vehicle Routing: Methods and Studies*, volume 16 of *Studies in Management Science and Systems*. North-Holland, 1988.

[Asc95] N. Ascheuer. *Hamiltonian Path Problems in the On-line Optimization of Flexible Manufacturing Systems*. PhD thesis, Technische Universität Berlin, 1995.

[Bal85] E. Balas. On the facial structure of scheduling polyhedra. *Mathematical Programming Study*, 24:179–218, 1985.

[BFP95] E. Balas, M. Fischetti, and W. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68:241–265, 1995.

[DDSS95] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Handbooks in Operations Research and Management Science*, volume 8, pages 35–139, 1995.

[DL91] M. Desrochers and G. Laporte. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 10:27–36, 1991.

[GH85] M. Grötschel and O. Holland. Solving matching problems with linear programming. *Mathematical Programming*, 33:243–259, 1985.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer Verlag, Berlin, 1988.

[GP85] M. Grötschel and M. Padberg. Polyhedral theory. In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors, *The Traveling Salesman Problem*, chapter 8. John Wiley & Sons Ltd., 1985.

[Grö77] M. Grötschel. *Polyedrische Charakterisierungen kombinatorischer Optimierungsprobleme*. Hain, Meisenheim am Glan, 1977.

[Grö80] M. Grötschel. On the symmetric traveling salesman problem: Solution of a 120-city problem. *Math. Progr. Studies*, 12:61–77, 1980.

[Grö94] M. Grötschel. Kombinatorische Optimierung. Manuskript einer im WS 1994/95 an der TU Berlin im Fachbereich Mathematik gehaltenen Vorlesung, 1994.

[JRT94] M. Jünger, G. Reinelt, and S. Thienel. Provably good solutions for the traveling salesman problem. *Mathematical Methods of Operations Research*, 40:183–217, 1994.

[JRT95] M. Jünger, G. Reinelt, and S. Thienel. Practical problem solving with cutting plane algorithms in combinatorial optimization. In L. Lovász W. Cook and P. Seymour, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 20, pages 111–152, 1995.

[Jun90] D. Jungnickel. *Graphen, Netzwerke und Algorithmen*. BI-Wiss.-Verl., Mannheim, Wien, Zürich, 1990.

[MTZ60] C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulations and traveling salesman problems. *J. Assoc. Comput. Mach.*, 7:326–329, 1960.

[Pad75] M. Padberg. A note on zero-one programming. *Operations Research*, 23:833–837, 1975.

[PR90] M. Padberg and G. Rinaldi. Facet identification for the symmetric traveling salesman polytope. *Mathematical Programming*, 47:219–257, 1990.

[PR91] M. Padberg and G. Rinaldi. A branch and cut algorithm for the resolution of large–scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

[Rei91] G. Reinelt. TSPLIB - a travelling salesman problem library. *ORSA Journal on Computing*, 3:376–384, 1991.

[Sav85] M.W.P. Savelsbergh. Local search for routing problems with time windows. *Annals of Operations Research*, 4:285–305, 1985.

[Sch89] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons Ltd., Chichester, 1989.

[vE95] C.A. van Eijl. A polyhedral approach to the delivery man problem. Technical Report COSOR 95-19, Eindhoven University of Technology, Department of Mathematics and Computing Science, Eindhoven, The Netherlands, 1995.