Technische
Universität
Darmstadt

Diskrete Optimierung

Nach einer Vorlesung vom Sommersemester 2006
gehalten von

Prof. Dr. Alexander Martin

Inhaltsverzeichnis

Vorwort

Dieses Vorlesungsskript basiert auf dem Skript zur Vorlesung *Diskrete Optimierung I* gehalten im Sommersemester 2001 und 2003.
Besonderer Dank bei der Verfassung des Skripts gilt Markus Möller, der das Teile des Skripts geschrieben und Korrektur gelesen hat. Dennoch ist das Skript noch nicht vollständig überarbeitet und es ist sicherlich nicht frei von Fehlern. Für Hinweise auf solche bin ich immer dankbar.

Darmstadt, April 2006 Prof. Dr. Alexander Martin

Kapitel 1

Einführung

Schwerpunkt der Vorlesung ist die Lösung von gemischt-ganzzahligen Programmen.

Definition 1.1 *Gemischt-Ganzzahliges Lineares Programm (engl. Mixed Integer Programm (MIP)):*
Gegeben $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $p \in \{0, 1, \ldots, n\}$.
Dann heißt

$$\begin{aligned} \max \quad & c^\top x \\ & Ax \leq b \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

gemischt-ganzzahliges lineares Programm (Optimierungsproblem).

Spezialfälle
$p = 0$: Lineares Programm (vgl. Einführung in die Optimierung)
$p = n$: (rein) ganzzahliges (lineares) Programm
$p = n$ und $x \in \{0, 1\}^n$: binäres (lineares) Programm oder 0/1 Programm.

In engem Zusammenhang zu MIPs stehen lineare kombinatorische Optimierungsprobleme, die uns auch in dieser Vorlesung interessieren werden.

Definition 1.2 *(lineares) kombinatorisches Optimierungsproblem.*
Gegeben ist eine endliche Grundmenge E, eine Teilmenge $\mathcal{I}$ der Potenzmenge 2^E von E (die Elemente in $\mathcal{I}$ heißen zulässige Mengen *oder* Lösungen*) und eine Funktion $c : E \to \mathbb{R}$. Für jede Menge $F \subseteq E$ definieren wir $c(F) = \sum_{e \in F} c_e$.*
Das Problem

$$\max_{I \in \mathcal{I}} c(I) \quad \textit{bzw.} \quad \min_{I \in \mathcal{I}} c(I)$$

heißt (Lineares) kombinatorisches Optimierungsproblem.

Bemerkung 1.3 *Kombinatorische Optimierungsprobleme können als lineare Programme modelliert werden.*

Beweis. Gegeben ein kombinatorisches Optimierungsproblem $(E, \mathcal{I}, c)$.
Definiere den Inzidenzvektor $X^F \in \mathbb{R}^E$ einer Menge $F \subseteq E$ durch

$$X_e^F = \begin{cases} 1 & \text{, falls } e \in F, \\ 0 & \text{, sonst,} \end{cases}$$

und setze $X := \left\{X^I \mid I \in \mathcal{I}\right\}$. Dann gilt

$$\max_{I \in \mathcal{I}} c(I) = \max \left\{c^\top x \mid x \in X\right\}.$$

Da c linear ist, gilt

$$\max \left\{c^\top x \mid x \in X\right\} = \max \left\{c^\top x \mid x \in \text{conv (X)}\right\}$$

Da X endlich ist, wie wir in Kapitel 1.2.1 sehen werden, conv (X) ein Polyeder P, d. h. es existiert eine Matrix $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ mit conv (X) = P(A, b).
In anderen Worten

$$\max_{I \in \mathcal{I}} c(I) = \max \left\{c^\top x \mid x \in P(A, b)\right\},$$

also ein lineares Programm. □

Wir werden in Kapitel 2 sehen, dass auch MIPs als lineare Programme aufgefasst werden können.
Lineare Programme können effizient gelöst werden, vgl. die Vorlesung „Einführung in die Optimierung“. Also, wo liegt das Problem?

1.1 Beispiele und Formulierungen

In der Vorlesung „Einführung in die Optimierung“ haben wir bereits viele Problembeispiele der ganzzahligen und kombinatorischen Optimierung kennengelernt. Wir wiederholen hier nochmals kurz die wichtigsten und geben ein/zwei Beispiele aus realen Anwendungen.

Beispiel 1.4 Das Zuordnungsproblem
In einer Firma gibt es n Mitarbeiter, die n Jobs ausführen können. Jede Person kann genau einen Job durchführen. Die Kosten, die der Firma entstehen, wenn Mitarbeiter i Job j ausführt, betragen c_{ij}. Wie soll die Firma ihre Mitarbeiter kostengünstig einsetzen:
Variablen:

$$x_{ij} = \begin{cases} 1, & \textit{falls Mitarbeiter } i \textit{ Job } j \textit{ bearbeitet,} \\ 0, & \textit{sonst.} \end{cases}$$

Binäres Programm:

$$\begin{aligned}
&\min \sum_{i,j} c_{ij} x_{ij} \\
&\sum_{j=1}^{n} x_{ij} = 1, \quad \textit{für alle } i = 1, \ldots, n \\
&\sum_{i=1}^{n} x_{ij} = 1, \quad \textit{für alle } j = 1, \ldots, n \\
&x_{ij} \in \{0,1\}, \quad \textit{für alle } i, j = 1, \ldots, n.
\end{aligned}$$

Beispiel 1.5 Das Rucksack-Problem (engl. knapsack)
Eine Firma möchte ein Budget von b Geldeinheiten in n Projekte investieren. Jedes dieser Projekte j kostet a_j Geldeinheiten und hat einen erwarteten Gewinn von c_j Geldeinheiten. In welche Projekte soll die Firma investieren?
Variablen:

$$x_j = \begin{cases} 1, & \textit{falls Projekt } j \textit{ gewählt wird,} \\ 0, & \textit{sonst.} \end{cases}$$

$0/1$ *Programm:*

$$\begin{aligned}
&\max \sum_{j=1}^{n} c_j x_j \\
&\sum_{j=1}^{n} a_j x_j \leq b \\
&x_j \in \{0,1\}, \ \textit{für } j = 1, \ldots, n.
\end{aligned}$$

Beispiel 1.6 Set-Packing | Partitioning | Covering Probleme
Gegeben eine endliche Grundmenge $E = \{1, \ldots, m\}$ und eine Liste $F_j \subseteq E, j = 1, \ldots, n$ von Teilmengen von E. Mit jeder Teilmenge F_j sind Kosten / Erträge c_j assoziiert. Das Problem eine bzgl. c minimale / maximale Auswahl von Teilmengen F_j zu finden, so dass jedes Element $e \in E$ in höchstens, genau, mindestens einer Teilmenge vorkommt, heißt Set-Packing, -partitioning *bzw.* -covering Problem.

Formulierung als $0/1$ Programm:
Variablen:

$$x_j = \begin{cases} 1, & \textit{falls } F_j \textit{ gewählt wird,} \\ 0, & \textit{sonst.} \end{cases}$$

Definiere zusätzlich eine $0/1$ Matrix $A \in \{0,1\}^{m \times n}$, wobei $a_{ij} = 1$ genau dann, wenn Teilmenge F_j Element $i \in E$ enthält.

0/1 *Programm:*

$$\min c^\top x$$

$$Ax \left\{ \begin{array}{c} \leq \\ = \\ \geq \end{array} \right\} \mathbb{1}$$

$$x \in \{0,1\}^n .$$

Beispiel 1.7 Optimale Steuerung von Gasnetzwerken

Beispiel 1.8 Frequenzplanung in der Telekommunikation

1.2 Grundlagen der Polyedertheorie

In der Vorlesung „Einführung in die Optimierung“ haben wir schon einige grundlegende Begriffe und Sätze zu Polyedern kennengelernt. In diesem Kapitel wollen wir unseren Erfahrungsschatz ein wenig erweitern und weiteref interessante Konzepte kennenlernen, die wir in den folgenden Kapiteln öfters brauchen werden.

1.2.1 Projektion von Polyedern

Projektionen von Polyedern werden uns künftig häufiger begegnen. Sie sind ein wichtiges Hilfsmittel, um gewisse Aussagen über Polyeder abzuleiten oder beweisen zu können. Wir wollen in diesem Abschnitt ein Verfahren angeben, wie man die Projektion eines Polyeders auf eine andere Menge berechnen kann und daraus dann weitere Konsequenzen ableiten.

Definition 1.9 *Es seien S, $H \subseteq \mathbb{K}^n$ und $c \in \mathbb{K}^n \setminus \{0\}$. Die Menge*

$$\{x \in H \mid \exists \lambda \in \mathbb{K} \text{ mit } x + \lambda c \in S\}$$

heißt Projektion von S auf H entlang c. *Gilt*

$$H = \{x \in \mathbb{K}^n \mid c^T x = \gamma\}$$

für ein $\gamma \in \mathbb{K}$, so heißt die Projektion orthogonal.

Wir interessieren uns für den Fall, dass S ein Polyeder ist, d.h. wir müssen uns überlegen, wie die Projektion von Halbräumen aussieht. Dazu machen wir zunächst eine einfache Beobachtung.

Bemerkung 1.10 *Sei $H \subseteq \mathbb{K}^n$, $c \in \mathbb{K}^n \setminus \{0\}$ und $S = \{x \in \mathbb{K}^n \mid a^T x \leq \alpha\}$ ein Halbraum sowie P die Projektion von S auf H entlang c. Dann gilt:*

(a) Ist a orthogonal zu c, dann gilt $P = H \cap S$.

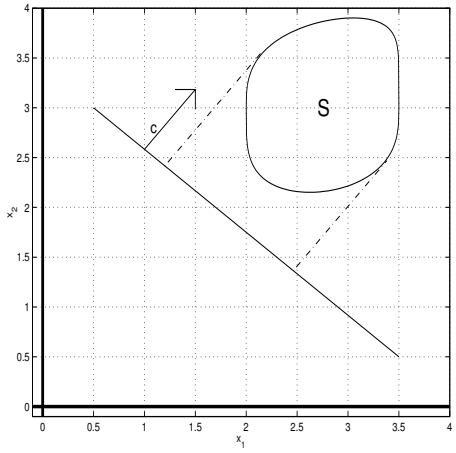

Abbildung 1.1: Projektion der Menge S auf H (Definition 1.9).

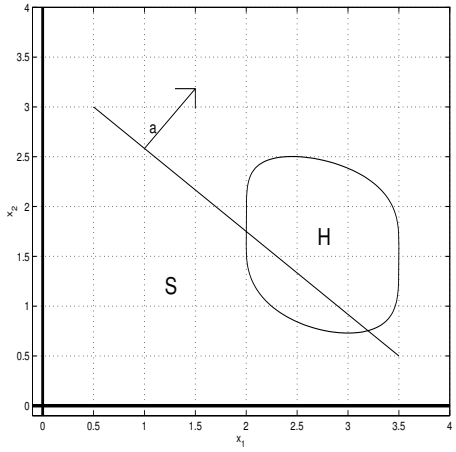

Abbildung 1.2: Projektion der Menge S auf H (Bemerkung 1.10).

(b) Ist a nicht orthogonal zu c, dann gilt $P = H$.

Beweis. Per Definition ist $P = \{x \in H \mid \exists\lambda \in \mathbb{K} \text{ mit } a^T(x + \lambda c) \leq \alpha\}$ und somit gilt in (a) die Hinrichtung „$\subseteq$“ per Definition; für die Rückrichtung wähle $\lambda = 0$. In (b) gilt die Hinrichtung „$\subseteq$“ ebenfalls per Definition, und für die Rückrichtung wähle $\lambda = \frac{\alpha - a^T x}{a^T c}$. □

Betrachten wir die Projektion des Durchschnitts zweier Halbräume

$$H_1 = \{x \in \mathbb{K}^n \mid a_1^T x \leq \alpha_1\},$$
$$H_2 = \{x \in \mathbb{K}^n \mid a_2^T x \leq \alpha_2\}.$$

Steht a_1 senkrecht zu c oder a_2 senkrecht zu c, so erhalten wir die Projektion aus der Beobachtung 1.10. Es bleiben die beiden folgenden Fälle zu unterscheiden:

(1) Beide Winkel $\measuredangle(a_i, c)$ sind kleiner als 90° oder beide Winkel liegen zwischen 90° und 180°. In diesem Fall ist die Projektion des Durchschnitts der beiden

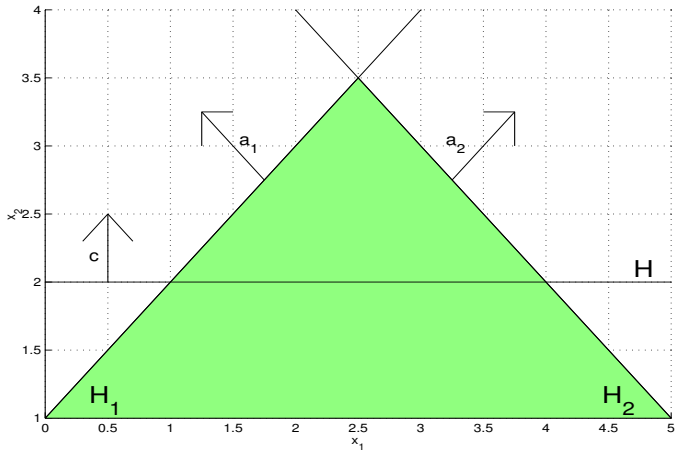

Abbildung 1.3: Winkel im Fall (1)

Halbräume wieder H.

(2) Ein Winkel ist kleiner als 90° und einer liegt zwischen 90° und 180°. In diesem Fall ist die Projektion von $H_1 \cap H_2$ auf H eine echte Teilmenge von H.

Wir wollen diese Teilmenge zunächst *geometrisch* bestimmen. Man betrachte dazu den Durchschnitt der zu H_1 und H_2 gehörenden Hyperebenen

$$H'_{12} \;=\; \{x \in \mathbb{K}^n \;\big|\; a_1^T x \;=\; \alpha_1 \;\text{ und }\; a_2^T x \;=\; \alpha_2\},$$

dann ist

$$G' \;=\; \{x \in \mathbb{K}^n \;\big|\; \exists y \in H'_{12} \text{ und } \lambda \in \mathbb{K} \text{ mit } x \;=\; y + \lambda c\}$$

eine Hyperebene, deren Normalenvektor senkrecht auf c steht. Sei G der zu G' gehörende Halbraum, der $H_1 \cap H_2$ enthält. Dann ist $G \cap H$ die gesuchte Projektion (vgl. Beobachtung 1.10 (a)).

Algebraisch lässt sich G wie folgt bestimmen: Aufgrund der Formel

$$\cos \measuredangle(x, y) \;=\; \frac{x^T y}{||x|| \cdot ||y||}$$

lassen sich Aussagen über den Winkel $\measuredangle(x, y)$ zwischen den Vektoren x und y beschreiben, wobei $x, y \neq 0$ gelte. Sei nun

$$\begin{aligned} 90^\circ \;&<\; \measuredangle(c, a_1) \;<\; 180^\circ \quad \text{und} \\ 0^\circ \;&<\; \measuredangle(c, a_2) \;<\; 90^\circ. \end{aligned}$$

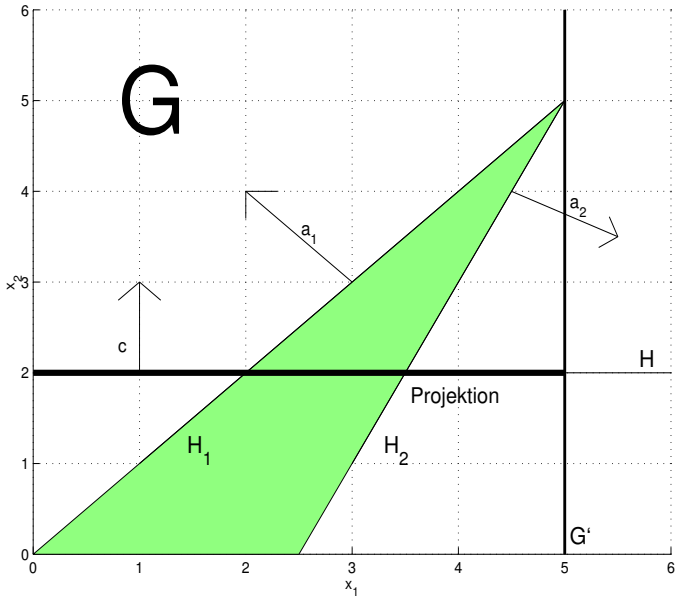

Abbildung 1.4: Geometrische Bestimmung der Projektion im Fall (2).

Dann ist $c^T a_1 < 0$ und $c^T a_2 > 0$. Wir bestimmen eine geeignete nicht-negative Linearkombination d aus a_1 und a_2 (genaue Formel siehe Satz 1.11), so dass d auf c senkrecht steht, d.h. $d^T c = 0$ gilt. Dann ist

$$G \;=\; \{x \in \mathbb{K}^n \;\big|\; d^T x \;\leq\; \delta\},$$

wobei δ entsprechend aus a_1 und a_2 erzeugt wird, der oben beschriebene Halbraum.

Die genauen Formeln und deren Korrektheit werden dargestellt im

Satz 1.11 *Sei $c \in \mathbb{K}^n \setminus \{0\}$, $H \subseteq \mathbb{K}^n$ eine beliebige Menge und*

$$H_i \;=\; \{x \in \mathbb{K}^n \;\big|\; a_i^T x \;\leq\; \alpha_i\}$$

für $i = 1, 2$. Wir bezeichnen mit P_H die Projektion von $H_1 \cap H_2$ auf H entlang c. Dann gelten die folgenden Aussagen:

(a) Gilt $a_i^T c = 0$ für $i = 1, 2$, so ist $P_H = H_1 \cap H_2 \cap H$.

(b) Gilt $a_1^T c = 0$ und $a_2^T c \neq 0$, so ist $P_H = H_1 \cap H$.

(c) Gilt $a_i^T c > 0$ für $i = 1, 2$ oder $a_i^T c < 0$ für $i = 1, 2$, so ist $P_H = H$.

(d) Gilt $a_1^T c < 0$ und $a_2^T c > 0$, dann setzen wir

$$\begin{aligned} d \;&=\; (a_2^T c) a_1 \;-\; (a_1^T c) a_2\,, \\ \delta \;&=\; (a_2^T c) \alpha_1 \;-\; (a_1^T c) \alpha_2\,, \end{aligned}$$

woraus dann die Darstellung

$$P_H \;=\; H \cap \{x \in \mathbb{K}^n \;\big|\; d^T x \;\leq\; \delta\}.$$

folgt.

Beachte in (d), dass $d^T x \leq \delta$ eine nicht-negative Linearkombination aus $a_i^T x \leq \alpha_i$ ist, und d auf c senkrecht steht. Zum Beweis des Satzes 1.11 benötigen wir folgendes Lemma.

Lemma 1.12 *Sei* $\bar{x} \in \mathbb{K}^n$, $c \in \mathbb{K}^n \setminus \{0\}$ *und* $H = \{x \in \mathbb{K}^n \mid a^T x \leq \alpha\}$ *mit* $a^T c \neq 0$. *Dann gilt:*

(i) $a^T c > 0 \quad \Rightarrow \quad \bar{x} + \lambda c \in H \quad \forall\, \lambda \leq \dfrac{\alpha - a^T \bar{x}}{a^T c},$

(ii) $a^T c < 0 \quad \Rightarrow \quad \bar{x} + \lambda c \in H \quad \forall\, \lambda \geq \dfrac{\alpha - a^T \bar{x}}{a^T c}.$

Beweis. Folgende Rechnung gilt sowohl für (i) als auch für (ii):

$$a^T(\bar{x} + \lambda c) \;=\; a^T \bar{x} \;+\; \lambda a^T c \;\leq\; a^T \bar{x} \;+\; \frac{\alpha - a^T \bar{x}}{a^T c} \cdot a^T c \;=\; \alpha.$$

□

Mit dieser Hilfe folgt nun der

Beweis von Satz 1.11.

(a),(b) beweist man analog zu Bemerkung 1.10.

(c) Da $P_H \subseteq H$ per Definition gilt, bleibt $H \subseteq P_H$ zu zeigen. Sei also $\bar{x} \in H$ beliebig. Setze

$$\bar{\lambda} \;=\; \min \left\{ \frac{\alpha_i - a_i^T \bar{x}}{a_i^T c} \;\Big|\; i = 1, 2 \right\}.$$

Dann gilt mit Lemma 1.12: $\bar{x} + \lambda c \in H_1 \cap H_2$ für alle $\lambda \leq \bar{\lambda}$. Daraus folgt nun mit Definition 1.9 die verlangte Aussage $\bar{x} \in P_H$.

(d) Sei

$$Q \;=\; \{x \in \mathbb{K}^n \;\big|\; d^T x \leq \delta\}$$

und Q_H die Projektion von Q auf H entlang c. Da $d^T x \leq \delta$ eine konische Kombination aus $a_i^T x \leq \alpha_i$ für $i = 1, 2$ ist, gilt also $H_1 \cap H_2 \subseteq Q$ und damit $P_H \subseteq Q_H$. Da weiterhin $d^T c = 0$ ist, gilt nach Bemerkung 1.10 entsprechend $Q_H = Q \cap H$ und damit $P_H \subseteq Q \cap H$. Es bleibt noch $Q \cap H \subseteq P_H$ zu zeigen. Dazu sei $\bar{x} \in Q \cap H$ beliebig und

$$\lambda_i \;=\; \frac{\alpha_i - a_i^T \bar{x}}{a_i^T c}, \text{ für } i = 1, 2.$$

Nach Voraussetzung ist

$$(a_2^T c)a_1^T \bar{x} \; - \; (a_1^T c)a_2^T \bar{x} \; = \; d^T \bar{x} \; \leq \; \delta \; = \; (a_2^T c)\alpha_1 \; - \; (a_1^T c)\alpha_2$$

und damit

$$(a_1^T c)(\alpha_2 - a_2^T \bar{x}) \; \leq \; (a_2^T c)(\alpha_1 - a_1^T \bar{x}) \qquad \Longleftrightarrow \quad \lambda_2 \; \geq \; \lambda_1.$$

Für ein beliebiges $\lambda \in [\lambda_1, \, \lambda_2]$ gilt nun

$$a_i^T(\bar{x} + \lambda c) \; = \; a_i^T \bar{x} \; + \; \lambda a_i^T c \; \leq \; a_i^T \bar{x} \; + \; \lambda_i a_i^T c \; = \; \alpha_i.$$

Also ist $\bar{x} + \lambda c \in H_1 \cap H_2$ und damit $\bar{x} \in P_H$.

□

Damit können wir unseren Projektionsalgorithmus angeben.

Algorithmus 1.13 *Projektion eines Polyeders entlang c.*

Input:

- $c \in \mathbb{K}^n$*: die Projektionsrichtung.*
- $P(A, b)$*: ein Polyeder.*

Output: $P(D, d)$*, so dass für jede beliebige Menge* $H \subseteq \mathbb{K}^n$ *die Menge*

$$H \; \cap \; P(D, d)$$

die Projektion von $P(A, b)$ *auf* H *entlang* c *ist.*

(1) *Partitioniere die Zeilenindexmenge* $M = \{1, 2, \ldots, m\}$ *von* A *wie folgt:*

$$\begin{aligned} N \; &= \; \{i \in M \; \big| \; A_{i\cdot} \, c \; < \; 0\}, \\ Z \; &= \; \{i \in M \; \big| \; A_{i\cdot} \, c \; = \; 0\}, \\ P \; &= \; \{i \in M \; \big| \; A_{i\cdot} \, c \; > \; 0\}. \end{aligned}$$

(2) *Setze* $r \; = \; |Z \cup (N \times P)|$ *und sei* $p : \{1, 2, \ldots, r\} \; \rightarrow \; Z \cup (N \times P)$ *eine Bijektion.*

(3) *Für* $i = 1, 2, \ldots, r$ *führe aus:*

(a) Ist $p(i) \in Z$*, dann setze* $D_{i\cdot} = A_{p(i)\cdot}$, $d_i = b_{p(i)}$ *(vgl. Satz 1.11 (a)).*

(b) Ist $p(i) = (s, t) \in N \times P$*, dann setze*

$$\begin{aligned} D_{i\cdot} \; &= \; (A_{t\cdot} \, c)A_{s\cdot} \; - \; (A_{s\cdot} \, c)A_{t\cdot}, \\ d_i \; &= \; (A_{t\cdot} \, c)b_s \; - \; (A_{s\cdot} \, c)b_t. \end{aligned}$$

(vgl. Satz 1.11 (d)).

(4) Stop *($P(D,d)$ ist das gesuchte Objekt).*

Satz 1.14 *Seien A, b, c, sowie D, d wie im Algorithmus 1.13 gegeben. H sei eine beliebige Menge und P_H die Projektion von $P(A,b)$ auf H entlang c. Dann gilt:*

(a) Für alle $i \in \{1, 2, \ldots, r\}$ existiert ein $u_i \geq 0$ mit $D_{i\cdot} = u_i^T A$, $d_i = u_i^T b$.

(b) Für alle $i \in \{1, 2, \ldots, r\}$ gilt $D_{i\cdot}\, c = 0$.

(c) $P_H = H \cap P(D,d)$.

(d) Sei $\bar{x} \in H$ und

$$\lambda_i = \frac{b_i - A_{i\cdot}\,\bar{x}}{A_{i\cdot}\,c} \quad \forall\, i \in P \cup N,$$

$$L = \begin{cases} -\infty, & \text{falls } N = \emptyset \\ \max\{\lambda_i \mid i \in N\}, & \text{falls } N \neq \emptyset. \end{cases}$$

$$U = \begin{cases} \infty, & \text{falls } P = \emptyset \\ \min\{\lambda_i \mid i \in P\}, & \text{falls } P \neq \emptyset. \end{cases}$$

Dann gilt:

(d1) $\bar{x} \in P(D,d) \Rightarrow L \leq U$ und $\bar{x} + \lambda c \in P(A,b)\ \forall\, \lambda \in [L,U]$.

(d2) $\bar{x} + \lambda c \in P(A,b) \Rightarrow \lambda \in [L,U]$ und $\bar{x} \in P(D,d)$.

Beweis.

(a) und (b) folgen direkt aus der Konstruktion von D.

(c) Zum Beweis dieses Punktes benutzen wir (d):
Sei $\bar{x} \in P_H$. Daraus folgt $\bar{x} \in H$ und die Existenz eines $\lambda \in \mathbb{K}$ mit $\bar{x} + \lambda c \in P(A,b)$. Mit (d2) folgt daraus $\bar{x} \in P(D,d)$.

Umgekehrt sei $\bar{x} \in H \cap P(D,d)$. Daraus folgt mit (d1) $\bar{x} + \lambda c \in P(A,b)$ für ein $\lambda \in \mathbb{K}$, also ist $\bar{x} \in P_H$.

(d1) Sei $\bar{x} \in P(D,d)$. Wir zeigen zuerst $L \leq U$.
Ist $P = \emptyset$ oder $N = \emptyset$, so ist dies offensichtlich richtig. Andernfalls sei $p(v) = (s,t)$ mit $\lambda_s = L$ und $\lambda_t = U$. Dann folgt analog zum Beweis von Satz 1.11 (d) die Behauptung $U \geq L$. Wir zeigen nun $A_{i\cdot}(\bar{x} + \lambda c) \leq b_i$ für alle $\lambda \in [L,U]$, $i \in P \cup N \cup Z$.

Ist $i \in Z$, so gilt mit $p(j) = i$: $D_{j\cdot} = A_{i\cdot}$, $d_j = b_i$. Aus $A_{i\cdot}\, c = 0$ folgt

$$A_{i\cdot}(\bar{x} + \lambda c) = A_{i\cdot}\,\bar{x} = D_{j\cdot}\,\bar{x} \leq d_j = b_i.$$

Ist $i \in P$, dann ist $U < +\infty$ und

$$\begin{aligned} A_{i\cdot}(\bar{x} + \lambda c) &= A_{i\cdot}\,\bar{x} + \lambda A_{i\cdot}\,c \leq A_{i\cdot}\,\bar{x} + U A_{i\cdot}\,c \\ &\leq A_{i\cdot}\,\bar{x} + \lambda_i A_{i\cdot}\,c = b_i \end{aligned}$$

Ist $i \in N$, so folgt die Behauptung entsprechend.

(d2) Sei $\bar{x}+\lambda c \in P(A,b)$. Angenommen, es gilt $\lambda \notin [L,U]$, wobei o.B.d.A. $\lambda < L$ angenommen werden kann. Dann gilt für $i \in N$ mit $\lambda_i > \lambda$:

$$A_{i\cdot}(\bar{x} + \lambda c) = A_{i\cdot}\,\bar{x} + \lambda A_{i\cdot}\,c > A_{i\cdot}\,\bar{x} + \lambda_i A_{i\cdot}\,c = b_i.$$

Dies ist ein Widerspruch. Es bleibt noch $\bar{x} \in P(D,d)$ zu zeigen.

Aus (a) und $A(\bar{x} + \lambda c) \leq b$ folgt $D(\bar{x} + \lambda c) \leq d$.

Nach (b) ist $Dc = 0$ und somit $D(\bar{x}+\lambda c) = D\bar{x} \leq d$, also ist $\bar{x} \in P(D,d)$. □

Ein wichtiger Spezialfall von Algorithmus 1.13 ist der

Algorithmus 1.15 *Fourier-Motzkin-Elimination (der j-ten Variable).*

Input: *Wie in Algorithmus 1.13, jedoch mit folgenden Änderungen:*

- $c = e_j$
- $H = \{x \in \mathbb{K}^n \mid x_j = 0\} = \{x \in \mathbb{K}^n \mid c^T x = 0\}$

Output: *Wie in Algorithmus 1.13, wobei gilt:*

$$\{x \in \mathbb{K}^n \mid Dx \leq d,\ x_j = 0\}$$

ist die orthogonale Projektion von $P(A,b)$ auf H.

(1) *Spezialfall von 1.13:*

$$\begin{aligned} N &= \{i \in M \mid A_{ij} < 0\}, \\ Z &= \{i \in M \mid A_{ij} = 0\}, \\ P &= \{i \in M \mid A_{ij} > 0\}. \end{aligned}$$

(2) *Wie in 1.13.*

(3) *Für $i \in \{1, 2, \ldots, r\}$ führe aus:*

(a) Wie in 1.13,

(b) wie in 1.13 mit

$$\begin{aligned} D_{i\cdot} &= a_{tj}A_{s\cdot} - a_{sj}A_{t\cdot}, \\ d_i &= a_{tj}b_s - a_{sj}b_t. \end{aligned}$$

(4) *wie in 1.13.*

Mit Hilfe der Fourier-Motzkin-Elimination (FME) erhalten wir einen alternativen Beweis zum Farkas-Lemma, vgl. die Vorlesung „Einführung in die Optimierung“. Bevor wir diesen durchführen können, benötigen wir noch eine Folgerung aus Satz 1.14.

Folgerung 1.16 *Es gilt:* $P(A,b) \neq \emptyset \iff P(D,d) \neq \emptyset.$

Beweis. Die Projektion von $P(A,b)$ entlang c auf sich selbst ist $P(A,b)$. Mit Satz 1.14 (c) gilt damit $P(A,b) = P(A,b) \cap P(D,d)$. Gilt nun $P(D,d) = \emptyset$, so folgt $P(A,b) = \emptyset$. Ist dagegen $P(D,d) \neq \emptyset$, so ergibt Satz 1.14 (d1) $P(A,b) \neq \emptyset$. □

Wenden wir nun die Fourier-Motzkin-Elimination auf die erste Variable (d.h. entlang e_1) an, so gilt mit Satz 1.14 (a), (b) und Folgerung 1.16:

$$\begin{aligned} &(i) && D^1 e_1 = 0. \\ &(ii) && \text{Es existiert eine Matrix } U^1 \geq 0 \text{ mit } U^1 A = D^1,\ U^1 b = d^1. \\ &(iii) && P(A,b) = \emptyset \iff P(D^1,d^1) = \emptyset. \end{aligned} \tag{1.1}$$

Entsprechend können wir die Fourier-Motzkin-Elimination fortsetzen und die zweite Variable eliminieren. Wir erhalten im zweiten Schritt eine Matrix D^2 mit:

$$\begin{aligned} &(i) && D^2 e_2 = 0. \\ &(ii) && \text{Es gibt eine Matrix } \bar{U}^2 \geq 0 \text{ mit } \bar{U}^2 D^1 = D^2 \text{ und } \bar{U}^2 d^1 = d^2. \\ &(iii) && P(D^1,d^1) = \emptyset \iff P(D^2,d^2) = \emptyset. \end{aligned} \tag{1.2}$$

Aus (1.2) (ii) und (1.1) (i) folgt

$$D^2 e_1 = \bar{U}^2 D^1 e_1 = \bar{U}^2\, 0 = 0,$$

und mit $U^2 = \bar{U}^2 U^1$ folgt aus (1.1) (ii) und (1.2) (ii)

$$U^2 A = \bar{U}^2 U^1 A = \bar{U}^2 D^1 = D^2,$$

d.h. es gilt:

$$\begin{aligned} &(i) && D^2 e_1 = 0\,, \quad D^2 e_2 = 0. \\ &(ii) && \text{Es existiert eine Matrix } U^2 \geq 0 \text{ mit } U^2 A = D^2,\ U^2 b = d^2. \\ &(iii) && P(A,b) = \emptyset \iff P(D^2,d^2) = \emptyset. \end{aligned} \tag{1.3}$$

Dieser Prozess lässt sich fortsetzen und nach n-maliger Projektion erhalten wir:

$$\begin{array}{lll} & (i) & D^n e_j = 0 \quad \forall\, j \in \{1, 2, \ldots, n\}. \\ (1.4) & (ii) & \text{Es existiert eine Matrix } U^n \geq 0 \text{ mit } U^n A = D^n,\ U^n b = d^n. \\ & (iii) & P(A,b) = \emptyset \quad \Longleftrightarrow \quad P(D^n, d^n) = \emptyset. \end{array}$$

Aus (1.4) (i) folgt $D^n = 0$. Damit gilt auch $P(D^n, d^n) \neq \emptyset \iff d^n \geq 0$. Gibt es ein i mit $d_i^n < 0$, so existiert nach (1.4) (ii) ein Vektor $u \geq 0$ mit $u^T A = D_{i\cdot}^n = 0$ und $u^T b = d_i^n < 0$. Mit (1.4) (iii) hat damit genau eines der beiden folgenden Gleichungssysteme eine Lösung (vgl. die Vorlesung „Einführung in die Optimierung“):

$$Ax \leq b \qquad \dot{\bigvee} \qquad \begin{aligned} u^T A &= 0 \\ u^T b &< 0 \\ u &\geq 0. \end{aligned}$$

Hier folgen noch ein paar Konsequenzen aus Satz 1.14 und Algorithmus 1.13.

Folgerung 1.17

(a) *Ist $H = P(A', b')$ ein Polyeder, dann ist die Projektion von $P(A, b)$ auf H entlang c ein Polyeder.*

(b) *Die Projektion eines Polyeders $P(A, b) \subseteq \mathbb{K}^n$ auf $\mathbb{K}^k$ mit $k \leq n$ ist ein Polyeder.*

Beweis.

(a) Nach Satz 1.14 (c) gilt

$$P_H = P(A', b') \cap P(D, d), \text{ d.h. } P_H = P\left(\begin{pmatrix} A' \\ D \end{pmatrix}, \begin{pmatrix} b' \\ d \end{pmatrix}\right).$$

(b) Folgt direkt aus (a), da $\mathbb{K}^k$ ein Polyeder ist.

□

Bemerkung 1.18 *Die Projektion Q eines Polyeders $P(A, b) \subseteq \mathbb{K}^n$ auf $\mathbb{K}^k$ mit $k \leq n$ wird beschrieben durch*

$$Q = \left\{ x \in \mathbb{K}^k \;\middle|\; \exists\, y \in \mathbb{K}^r \text{ mit } \begin{pmatrix} x \\ y \end{pmatrix} \in P(A, b) \right\},$$

wobei $k + r = n$ gilt und o.B.d.A. x zu den ersten k Spalten von A gehört.

Beweis. Vergleiche Definition 1.9 und die Fourier-Motzkin-Elimination. □

Mit Hilfe der Projektion wollen wir noch einige Operationen nachweisen, die Polyeder erzeugen oder Polyeder in Polyeder überführen.

Eine affine Abbildung $f : \mathbb{K}^n \to \mathbb{K}^k$ ist gegeben durch eine Matrix $D \in \mathbb{K}^{k\times n}$ und einen Vektor $d \in \mathbb{K}^k$, so dass $f(x) = Dx + d$ für alle $x \in \mathbb{K}^n$ gilt.

Satz 1.19 *Affine Bilder von Polyedern sind Polyeder.*

Beweis. Sei $P = P(A, b) \subseteq \mathbb{K}^n$ ein Polyeder und $f : \mathbb{K}^n \to \mathbb{K}^k$ mit $f(x) = Dx + d$ eine affine Abbildung. Dann gilt

$$\begin{aligned} f(P) &= \{y \in \mathbb{K}^k \mid \exists\, x \in \mathbb{K}^n \text{ mit } Ax \le b \text{ und } y = Dx + d\} \\ &= \left\{y \in \mathbb{K}^k \mid \exists\, x \in \mathbb{K}^n \text{ mit } B\begin{pmatrix} x \\ y \end{pmatrix} \le \bar{b}\right\} \end{aligned}$$

mit

$$B = \begin{pmatrix} A & 0 \\ D & -I \\ -D & I \end{pmatrix} \qquad \text{und} \qquad \bar{b} = \begin{pmatrix} b \\ -d \\ d \end{pmatrix}.$$

Letzteres ist ein Polyeder nach Bemerkung 1.18 und Folgerung 1.17. □

Folgerung 1.20 (Satz von Weyl)
Für jede Matrix $A \in \mathbb{K}^{m\times n}$ gilt:

$$\left.\begin{array}{r} lin\,(A) \\ aff\,(A) \\ conv\,(A) \\ cone\,(A) \end{array}\right\} \quad \textit{ist ein Polyeder.}$$

Beweis.
Wir beweisen exemplarisch, dass cone (A) ein Polyeder ist, die anderen Fälle können analog dazu bewiesen werden. Es gilt:

$$\text{cone}\,(A) = \{x \in \mathbb{K}^m \mid \exists\, y \ge 0 \text{ mit } x = Ay\}.$$

Mit $P = P(-I, 0)$ und $f(x) = Ax$ gilt $f(P) = \text{cone}\,(A)$ und damit ist nach Satz 1.19 cone (A) ein Polyeder. □

Folgerung 1.21 *Die Summe $P_1 + P_2$ zweier Polyeder P_1 und P_2 ist wieder ein Polyeder.*

Beweis. Sei $P_i = P(A^i, b^i)\,,\ i = 1, 2$. Dann ist

$$\begin{aligned} P_1 + P_2 &= \{x_1 + x_2 \in \mathbb{K}^n \mid A^1 x_1 \leq b^1\,,\ A^2 x_2 \leq b^2\} \\ &= \{z \in \mathbb{K}^n \mid \exists\, x_1,\, x_2 \in \mathbb{K}^n :\ A^1 x_1 \leq b^1\,,\ A^2 x_2 \leq b^2\,,\ z = x_1 + x_2\}. \end{aligned}$$

Mit

$$P = P\left(\begin{pmatrix} A^1 & 0 \\ 0 & A^2 \end{pmatrix},\ \begin{pmatrix} b^1 \\ b^2 \end{pmatrix}\right) \quad \text{und} \quad f\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = Ix_1 + Ix_2$$

gilt $f(P) = P_1 + P_2$ und damit ist nach Satz 1.19 $f(P)$ wieder ein Polyeder. □

Verbinden wir die beiden Folgerungen 1.20 und 1.21, so erhalten wir

Folgerung 1.22 *Es seien $A \in \mathbb{K}^{m\times n}$ und $B \in \mathbb{K}^{m\times n'}$. Dann gilt:*

$$P = \mathit{conv}(A) + \mathit{cone}(B) \qquad \textit{ist ein Polyeder.}$$

Folgerung 1.22 eröffnet uns eine andere Sichtweise auf Polyeder. Wir werden im folgenden Abschnitt zeigen, dass sich jedes Polyeder in der Form $\mathrm{conv}\,(A) + \mathrm{cone}\,(B)$ schreiben läßt. Dazu bedarf es jedoch einiger Vorbereitungen.

1.2.2 Darstellungssätze

Betrachten wir nochmals das Farkas-Lemma aus einem anderen Blickwinkel:

$$\exists\, x \geq 0 :\ Ax = b \qquad \dot{\bigvee} \qquad \exists\, y :\ y^T A \leq 0\,,\ y^T b > 0$$

oder anders ausgedrückt:

$$\exists\, x \geq 0 :\ Ax = b \qquad \Longleftrightarrow \qquad \forall\, y :\ A^T y \leq 0 \ \Rightarrow\ y^T b \leq 0.$$

Damit sind alle rechten Seiten b charakterisiert, für die das lineare Programm $Ax = b\,,\ x \geq 0$ eine Lösung hat. Nach Definition gilt

$$\mathrm{cone}\,(A) \ = \ \{b \in \mathbb{K}^m \mid \exists\, x \geq 0 \text{ mit } Ax = b\}.$$

Zusammen mit dem Farkas-Lemma haben wir dann

Bemerkung 1.23 *Für alle Matrizen $A \in \mathbb{K}^{m\times n}$ gilt*

$$\mathit{cone}\,(A) \ = \ \{b \in \mathbb{K}^m \mid y^T b \leq 0 \ \forall\, y \in P(A^T, 0)\}.$$

Die geometrische Interpretation von Bemerkung 1.23 könnte man in der folgenden Form schreiben:

Die zulässigen rechten Seiten b von $Ax = b,\ x \geq 0$	$\hat{=}$	Die Vektoren, die mit allen Vektoren aus $P(A^T, 0)$ einen stumpfen Winkel bilden.

Als Verallgemeinerung geben wir

Definition 1.24 *Sei $S \subseteq \mathbb{K}^n$ eine beliebige Menge. Die Menge aller Vektoren, die einen stumpfen Winkel mit allen Vektoren aus S bilden, heißt* polarer Kegel. *In Zeichen:*

$$S^\circ \;=\; \{y \in \mathbb{K}^n \;\big|\; y^T x \leq 0 \;\forall\, x \in S\}.$$

Für eine Matrix A bedeutet A° die Menge $\{y \in \mathbb{K}^n \,|\, y^T A \leq 0\,\}$.

Bemerkung 1.23 lässt sich nun in der folgenden Form schreiben.

Folgerung 1.25 *Für alle Matrizen $A \in \mathbb{K}^{m\times n}$ gilt*

$$P(A,0)^\circ \;=\; \mathit{cone}\,(A^T).$$

Beispiel 1.26 *Sei*

$$A \;=\; \begin{pmatrix} -3 & 2 \\ 1 & -2 \end{pmatrix}.$$

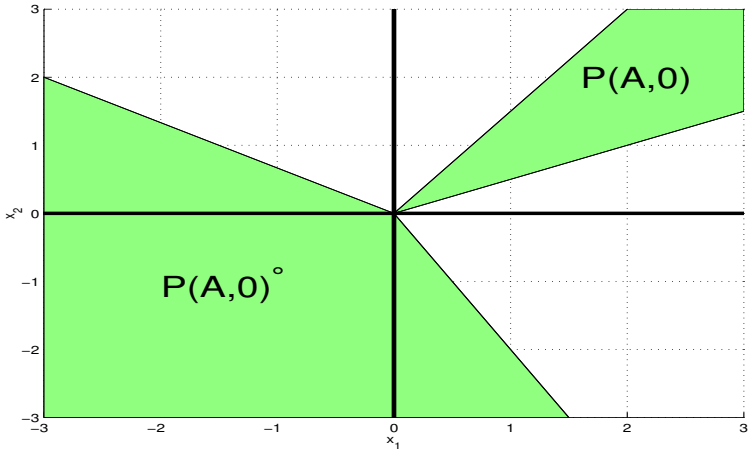

Abbildung 1.5: Zu Beispiel 1.26: $P(A,0)^\circ = \text{cone}\left(\{\binom{-3}{2}, \binom{1}{-2}\}\right)$.

Es gilt: $Ax = b,\ x \geq 0$ *ist lösbar* $\quad\Longleftrightarrow\quad b \in P(A^T,0)^\circ$

$$\Longleftrightarrow\quad b \in \mathit{cone}\,(A)$$

Zum Beispiel ist

$$\begin{cases} Ax = \binom{0}{1},\ x \geq 0 & \text{nicht lösbar.} \\ Ax = \binom{-1}{0},\ x \geq 0 & \text{lösbar.} \end{cases}$$

Wir schreiben im Weiteren kurz $S^{\circ\circ} \;=\; (S^\circ)^\circ$.

Lemma 1.27 *Für S, $S_i \subseteq \mathbb{K}^n$, $i \in \{1, 2, \ldots, k\}$ gilt:*

(a) $S_i \subseteq S_j \quad \Longrightarrow \quad S_j^\circ \subseteq S_i^\circ$.

(b) $S \subseteq S^{\circ\circ}$.

(c) $\left(\bigcup_{i=1}^{k} S_i\right)^\circ = \bigcap_{i=1}^{k} S_i^\circ$.

(d) $S^\circ = cone\,(S^\circ) = (cone\,(S))^\circ$.

Beweis. Übung. □

Folgerung 1.28

$$(cone\,(A^T))^\circ = cone\,(A^T)^\circ = P(A, 0).$$

Beweis. Es gilt:

$$(\text{cone}\,(A^T))^\circ \overset{1.27(d)}{=} \text{cone}\,(A^T)^\circ \overset{1.27(d)}{=} (A^T)^\circ \overset{1.24(a)}{=} \{x \mid Ax \leq 0\} = P(A, 0).$$

□

Satz 1.29 (Polarensatz)
Für jede Matrix A gilt

$$\begin{aligned} P(A, 0)^{\circ\circ} &= P(A, 0), \\ cone\,(A)^{\circ\circ} &= cone\,(A). \end{aligned}$$

Beweis. Es gilt

$$P(A, 0) \overset{1.28}{=} \text{cone}\,(A^T)^\circ \overset{1.25}{=} P(A, 0)^{\circ\circ},$$

$$\text{cone}\,(A) \overset{1.25}{=} P(A^T, 0)^\circ \overset{1.28}{=} (\text{cone}\,(A)^\circ)^\circ \overset{1.27}{=} \text{cone}\,(A)^{\circ\circ}.$$

□

Damit haben wir alle Hilfsmittel zusammen, um zu zeigen, dass Polyeder nicht nur in der Form $P(A, b)$ dargestellt werden können. Genau dies besagt der

Satz 1.30 (Satz von Minkowski)
Eine Teilmenge $K \subseteq \mathbb{K}^n$ ist genau dann ein polyedrischer Kegel, wenn K die konische Hülle endlich vieler Vektoren ist, d.h. zu jeder Matrix $A \in \mathbb{K}^{m \times n}$ gibt es eine Matrix $B \in \mathbb{K}^{n \times d}$ mit

$$P(A, 0) = cone\,(B)$$

und umgekehrt.

Beweis. Es gilt:

$$P(A,0) \overset{1.28}{=} \text{cone}\,(A^T)^\circ \overset{1.20}{=} P(B^T,0)^\circ \overset{1.25}{=} \text{cone}\,(B).$$

□

Satz 1.31 *Es seien $A \in \mathbb{K}^{m\times n}$, $b \in \mathbb{K}^m$. Dann existieren endliche Mengen $V, E \subseteq \mathbb{K}^n$ mit*

$$P(A,b) \;=\; conv\,(V) \;+\; cone\,(E).$$

Beweis. Setze

$$H \;=\; P\left(\begin{pmatrix} A & -b \\ 0^T & -1 \end{pmatrix},\; \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right).$$

Dann gilt: $x \in P(A,b) \;\Leftrightarrow \binom{x}{1} \in H$.

H ist ein polyedrischer Kegel, also gibt es nach Satz 1.30 eine Matrix $B \in \mathbb{K}^{(n+1)\times d}$ mit $H = \text{cone}\,(B)$. Aufgrund der Definition von H (vgl. die letzte Zeile) hat die letzte Zeile von B nur nicht-negative Einträge. Durch Skalierung und Vertauschung der Spalten von B können wir B in eine Matrix $\bar{B}$ überführen, für die dann gilt:

$$\bar{B} \;=\; \begin{pmatrix} V & E \\ \mathbb{1}^T & 0^T \end{pmatrix} \quad \text{mit } \text{cone}\,(\bar{B}) = H.$$

Damit gilt nun

$$\begin{aligned} x \in P(A,b) \quad &\Longleftrightarrow \quad \begin{pmatrix} x \\ 1 \end{pmatrix} \in H \\ &\Longleftrightarrow \quad \begin{array}{l} x \;=\; V\cdot\lambda \;+\; E\cdot\mu \\ 1 \;=\; \mathbb{1}^T\cdot\lambda \end{array} \quad \lambda,\, \mu \;\geq\; 0 \\ &\Longleftrightarrow \quad x \;\in\; \text{conv}\,(V) \;+\; \text{cone}\,(E). \end{aligned}$$

□

Folgerung 1.32 *Eine Teilmenge $P \subseteq \mathbb{K}^n$ ist genau dann ein Polytop, wenn P die konvexe Hülle endlich vieler Vektoren ist.*

Beweis. Sei $V \subseteq \mathbb{K}^n$ endlich und $P = \text{conv}\,(V)$, dann ist P nach Folgerung 1.20 ein Polyeder. Ist $x \in P$, so gilt

$$x \;=\; \sum_{i=1}^{k} \lambda_i v_i \;\text{ mit }\; v_i \in V\,,\; \lambda_i \geq 0\,,\; \sum_{i=1}^{k} \lambda_i = 1$$

und somit

$$||x|| \;\leq\; \sum_{i=1}^{k} ||v_i|| \quad \text{also} \quad P \;\subseteq\; \left\{ x \in \mathbb{K}^n \;\middle|\; ||x|| \;\leq\; \sum_{v\in V} ||v|| \right\}.$$

Also ist P beschränkt, d.h. P ist ein Polytop.

Umgekehrt, ist P ein Polytop, so gibt es nach Satz 1.31 endliche Mengen V, E mit $P = \text{conv}\,(V) + \text{cone}\,(E)$. Angenommen, es existiert ein $e \in E$ mit $e \neq 0$, so gilt $x + \nu e \in P$ für alle $\nu \in \mathbb{N}$ und alle $x \in \text{conv}\,(V)$. Demnach ist P unbeschränkt, falls $E \setminus \{0\} \neq \emptyset$. Daher muss $E \in \{\emptyset, \{0\}\}$ gelten und damit

$$\text{conv}\,(V) \;=\; \text{conv}\,(V) \;+\; \text{cone}\,(E) \;=\; P.$$

□

Satz 1.33 (Darstellungssatz)
Eine Teilmenge $P \subseteq \mathbb{K}^n$ ist genau dann ein Polyeder, wenn P die Summe eines Polytops und eines polyedrischen Kegels ist, d.h. wenn es endliche Mengen V, $E \subseteq \mathbb{K}^n$ gibt mit

$$P \;=\; \mathit{conv}\,(V) \;+\; \mathit{cone}\,(E).$$

Beweis. Kombiniere

Satz 1.30: polyedrischer Kegel $= \text{cone}\,(E)$,

Satz 1.31: $P(A,b) \;=\; \text{conv}\,(V) + \text{cone}\,(E)$,

Folgerung 1.32: Polytop $= \text{conv}\,(V)$,

Folgerung 1.22: $\text{conv}\,(V) + \text{cone}\,(E)$ ist ein Polyeder.

□

Damit kennen wir jetzt zwei Darstellungsformen für Polyeder

(1) Die äußere Beschreibung: $P(A,b)$

$$P(A,b) \;=\; \bigcap_{i=1}^{m} \{x \in \mathbb{K}^n \;\big|\; A_{i\cdot}\, x \;\leq\; b_i\} \;\subseteq\; \{x \;\big|\; A_{i\cdot}\, x \;\leq\; b_i\},$$

d.h. $P(A,b)$ wird als Durchschnitt von größeren Mengen (Halbräumen) betrachtet.

(2) Die innere Beschreibung: $\text{conv}\,(V) + \text{cone}\,(E)$

Ist $E = \emptyset$, so ist die Bezeichnung offensichtlich, denn es gilt $V \subseteq P$ und somit wird P durch eine konvexe Hüllenbildung aus Elementen von sich selbst erzeugt. Analoges gilt, wenn P ein polyedrischer Kegel ist. Gilt $V \neq \emptyset$ und $E \neq \emptyset$, so ist E nicht notwendigerweise Teilmenge von P, jedoch gelingt es immer, P durch Vektoren $v \in V$ und $e \in E$ „von innen heraus“ zu konstruieren, vgl. Abbildung 1.6.

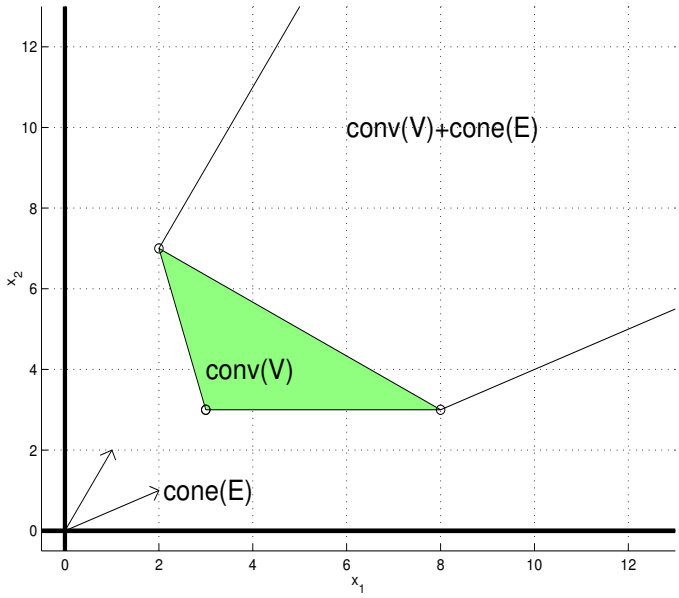

Abbildung 1.6: innere Konstruktion eines Polyeders.

Kapitel 2

Ganzzahlige Polyeder

In diesem Kapitel wollen wir uns zunächst mit Polyedern beschäftigen, die aus ganzzahliger Programme resultieren und deren Ecken alle ganzzahlig sind. Solle Situationen können als „Glücksfälle“ betrachtet werden, denn in diesen Fällen reicht es das zugrundeliegende lineare Programm zu lösen.

2.1 Zulässige Mengen ganzzahliger Programme

Die zulässigen Mengen von (gemischt-) ganzzahligen Programmen sind spezielle Teilmengen von Polyedern.
Intuitiv könnte man meinen, dass durch Bildung der konvexen Hülle aller zulässigen Punkte wieder ein Polyeder entsteht. Dem ist aber nicht so.

Beispiel 2.1 *Betrachte*

$$\begin{array}{rrcl} z^* = \max & -\sqrt{2}x & + & y \\ & -\sqrt{2}x & + & y \leq 0 \\ & x & \geq & 1 \\ & y & \geq & 0 \\ & & x, y & \textit{ganzzahlig} \end{array} \tag{2.1}$$

Dann gilt
(a) (2.1) hat zulässige Lösungen (bezeichne diese mit S).
(b) (2.1) ist nach oben beschränkt.
(c) (2.1) hat keine Optimallösung (sonst wäre $\sqrt{2}$ rational).
Wäre nun conv (S) ein Polyeder, so wäre

$$z^* = \max\left\{\sqrt{2}x + y \mid x \in S\right\} = \max\left\{\sqrt{2}x + y \mid x \in \text{conv } (S)\right\}$$

und besäße eine optimale Ecklösung, Widerspruch zu (3).
Der Grund für obige Tatsache liegt daran, dass wir irrationale Daten erlauben.

Definition 2.2 *Ein Polyeder $P \subseteq \mathbb{K}^n$ heißt* rational, *falls es eine Matrix $A \in \mathbb{Q}^{m\times n}$ und einen Vektor $b \in \mathbb{Q}^m$ gibt mit $P = P(A, b)$.*
Darüber hinaus bezeichnen wir mit $P_{I,p} := conv\,\{x \in P \mid x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}\}$ für $p \in \{0, 1, \ldots, n\}$, wobei wir auch P_I als Abkürzung für $P_{I,n}$ schreiben.

Zunächst zwei einfache Beobachtungen

Beobachtung 2.3

(a) Ist P ein beschränktes Polyeder, so ist P_I ein Polyeder.

(b) Ist P ein rationaler polyedrischer Kegel, so gilt $P = P_I$.

Beweis.

(a) Da P beschränkt, ist $X = \{x \in \mathbb{Z}^n \mid x \in P\}$ endlich. Nach Folgerung 1.20, ist conv $(X) = P_I$ ein Polyeder.

(b) Klar.

□

Bemerkung 2.4

(a) 2.3(a) gilt nicht nur für rationale Polyeder.

(b) Entsprechende Aussagen gelten auch für $P_{I,p}$ für alle $p = 0, 1, \ldots, n$.

Satz 2.5 *Ist P ein rationales Polyeder, dann ist P_I ebenfalls ein (rationales) Polyeder und es gilt $rec\,(P) = rec\,(P_I)$.*

Beweis. Aus Satz 1.33 wissen wir, P hat eine Darstellung der Form $P = Q+C$, wobei Q ein Polytop ist und C ein Kegel. Nach Beobachtung 2.3 (b) gibt es $y_i \in \mathbb{Z}^n, i = 1, \ldots, s$, mit $C = \text{cone}\,(\{y_i, \ldots, y_s\})$. Wir setzen

$$B := \left\{ \sum_{i=1}^{s} \mu_i y_i \mid 0 \le \mu_i \le 1, i = 1, \ldots, s \right\} .$$

Wir behaupten

$$P_I = (Q + B)_I + C\,.$$

Daraus folgt die Behauptung, denn $(Q+B)$ ist beschränkt, also nach Beobachtung 2.3 (a) ist $(Q+B)_I$ ein Polyeder und falls $P_I \neq \emptyset$, dann ist C der Rezessionskegel von P_I.

$\subseteq$: Sei $p \in P_I$ ein ganzzahliger Punkt in P.

Dann ex. $q \in Q$ und $c \in C$ mit $p = q + c$. Für c gilt $c = \sum_{i=1}^{s} \mu_i y_i$ mit $\mu_i \geq 0$.

Setze $c' = \sum_{i=1}^{s} \lfloor \mu_i \rfloor y_i$ und $b = c - c'$, so ist c' ganzzahlig und $q + b = p - c'$ ebenfalls, da p und c' ganzzahlig.

Außerdem ist $b \in B$ nach Definition und damit $q + b \in (Q + B)_I$. Also ist $p = q + b + c' \in (Q + B)_I + C$.

$\supseteq$: Hier gilt:

$$(Q + B)_I + C \subseteq P_I + C \underbrace{=}_{Beob.2.3\ (b)} P_I + C_I = (P + C)_I = P_I.$$

□

Entsprechend zeigt man

Satz 2.6 *Ist P ein rationales Polyeder, so ist $P_{I,p}$ ebenfalls ein rationales Polyeder für alle $p \in \{0, 1, \ldots, n\}$.*

Wir wissen nun also, dass sich jedes MIP auf ein lineares Programm zurückführen lässt, d.h. es ex. $D \in \mathbb{Q}^{m \times n}$, $d \in \mathbb{Q}^m$ mit $P_I = P(D, d)$. Wie sieht D, d aus? Zunächst eine Beobachtung, die wir jedoch nicht beweisen wollen.

Lemma 2.7 *Sei $P = P(A, b)$ ein rationales Polyeder, so dass die Kodierungslänge jeder Ungleichung höchstens φ ist.*

(a) Dann gibt es ein Ungleichungssystem $Dx \leq d$ mit $P_I = P(D, d)$, so dass die Kodierungslänge jeder Ungleichung höchstens $15n^6\varphi$ ist.

(b) Falls $P_I \neq \emptyset$, so gibt es einen ganzzahligen Punkt, dessen Kodierungslänge höchstens $5n^4\varphi$ ist.

Zum Beweis dieses Lemmas siehe beispielsweise [35], Kapitel 17.

Satz 2.8 *Das Entscheidungsproblem „Hat ein rationales Ungleichungssystem $Ax \leq b$ eine ganzzahlige Lösung?“ ist $\mathcal{NP}$-vollständig.*

Beweis. Zur Erinnerung: $\Pi \in \mathcal{NP} \Leftrightarrow$

(a) Für alle $I \in \Pi$, auf die die Antwort "JA" lautet, existiert ein Zertifikat, polynomial in $\langle I \rangle$, mit dessen Hilfe die Korrektheit der Antwort überprüft werden kann.

(b) Es existiert ein Algorithmus A mit Input I und Zertifikat z, polynomial in $\langle I \rangle$, der die Korrektheit der Antwort bestätigt.

$\Pi \in \mathcal{NP}$-schwer, falls ein polynomialer Algorithmus zu dessen Lösung verwendet werden kann, um jedes Problem in $\mathcal{NP}$ zu lösen. Π ist $\mathcal{NP}$-vollständig, falls es in $\mathcal{NP}$ ist und $\mathcal{NP}$-schwer.

Aus Lemma 2.7 folgt direkt, dass obiges Entscheidungsproblem in $\mathcal{NP}$ ist.

Zum Beweis der Vollständigkeit transformieren wir das Zulässigkeitsproblem (engl. Satisfiability Problem, kurz SAT) auf unser Entscheidungsproblem.

Definition 2.9 *Problem SAT*

Gegeben: *Logische Variablen $(x_1, \ldots, x_n)$, Menge von Klauseln $C_1, \ldots, C_m$, wobei jede Klausel aus der Disjunktion von Variablen x_j oder deren Negation $\bar{x}_j$ besteht.*

Frage: *Gibt es eine Belegung der Variablen mit TRUE und FALSE, so dass alle Klauseln wahr sind.*

Beispiel: $(x_1 \vee \bar{x}_2 \vee x_4) \wedge (x_2 \vee \bar{x}_3) \wedge (x_3 \vee \bar{x}_4)$.

Bemerkung 2.10 *SAT ist $\mathcal{NP}$-vollständig, [8].*

Modellierung als Ganzzahliges Programm

$$x_j = \begin{cases} 1, & \text{falls Literal } j \text{ TRUE ist,} \\ 0, & \text{falls Literal } j \text{ FALSE ist.} \end{cases}$$

$$\max\ 0^T x$$
$$\sum_{j: x_j \in C_i} x_j + \sum_{j: \bar{x}_j \in C_i} (1 - x_j) \geq 1$$
$$x_j \in \{0, 1\}$$

Die Transformation ist polynomial. Daraus folgt Satz 2.8 □

Da also das Finden einer zulässigen Lösung eines MIPs $\mathcal{NP}$-vollständig und damit das Lösen eines MIPs $\mathcal{NP}$-schwer ist, LPs jedoch in polynomialer Zeit gelöst werden können, können wir i.A. nicht damit rechnen, eine vollständige Beschreibung von P_I zu finden, bzw. das zugehörige Separierungsproblem polynomial zu lösen. Wir wollen uns in nächsten Kapitel zunächst mit Fällen beschäftigen, wo dies möglich ist, und später sehen, was man tun kann, wenn man diese Eigenschaft nicht hat.

Definition 2.11 *Ein rationales Polyeder P heißt* ganzzahlig, *falls $P = P_I$.*

Bemerkung 2.12 *Sei P ein rationales Polyeder. Dann sind äquivalent*

(a) $P = P_I$.

(b) Jede Seitenfläche von P enthält einen ganzzahligen Punkt.

(c) Jede minimale SF von P enthält einen ganzzahligen Punkt.

(d) $\max\{c^T x \mid x \in P\}$ *wird von einem ganzzahligen Punkt angenommen, für alle* $c \in \mathbb{R}^n$*, für die das Maximum endlich ist.*

Beweis. Übung. □

2.2 Totale Unimodularität

In diesem Kapitel untersuchen wir Eigenschaften der Nebenbedingungsmatrix A, so dass für jede rechte Seite b das Polyeder $\{x \mid Ax \leq b,\ x \geq 0\}$ ganzzahlig ist. Zur Motivation der Definition von (totaler) Unimodularität betrachte das Polyeder $\{x \geq 0 \mid Ax = b\}$, wobei A vollen Zeilenrang habe sowie A und b ganzzahlig seien. Zu jeder Ecke x gibt es eine Basis B, so dass $x_B = A_B^{-1} b$ und $x_N = 0$ gilt, vgl. die Vorlesung „Einführung in die Optimierung“. Für eine Matrix A mit $\det A_B = \pm 1$ stellt die Cramersche Regel sicher, dass A_B^{-1} ganzzahlig ist. Deshalb kann die Ganzzahligkeit von x_B und damit der Ecke x, garantiert werden, indem wir fordern, dass $\det A_B$ gleich ± 1 ist.

Definition 2.13

(a) Sei A eine $m \times n$ Matrix mit vollem Zeilenrang. A heißt unimodular*, falls alle Einträge von A ganzzahlig sind und jede invertierbare $m \times m$-Untermatrix von A Determinante ± 1 hat.*

(b) Eine Matrix A heißt total unimodular*, falls jede quadratische Untermatrix Determinante ± 1 oder 0 hat.*

Beachte, dass alle Einträge (die quadratischen Untermatrizen der Größe 1) einer total unimodularen Matrix entweder 0 oder ± 1 sind. Die folgenden Beobachtungen sind leicht einzusehen:

Beobachtung 2.14

(a) A ist genau dann total unimodular, wenn $[A, I]$ unimodular ist.

(b) A ist genau dann total unimodular, wenn $\begin{bmatrix} A \\ -A \\ I \\ -I \end{bmatrix}$ *total unimodular ist.*

(c) A ist genau dann total unimodular, wenn A^T total unimodular ist.

Beweis. Übung. □

Beispiel 2.15 *Sei A die Knoten-Kanten-Inzidenzmatrix eines gerichteten Graphen $G = (V, E)$ (d.h. $A = (a_{ij})_{i \in V, j \in E}$ mit $a_{ij} = 1$, falls $j \in \delta^+(i)$ und mit $a_{ij} = -1$, falls $j \in \delta^-(i)$ und $a_{ij} = 0$ sonst). Dann ist A total unimodular.*

Beweis. Übung. □

Die folgenden drei Sätze präzisieren, dass ein lineares Programm mit einer unimodularen bzw. total unimodularen Nebenbedingungsmatrix immer eine ganzzahlige Optimallösung besitzt, sofern das Optimum endlich ist.

Satz 2.16 *Sei A eine ganzzahlige $m \times n$ Matrix mit vollem Zeilenrang. Dann ist das Polyeder $\{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$ genau dann ganzzahlig für alle ganzzahligen Vektoren $b \in \mathbb{Z}^m$, wenn A unimodular ist.*

Beweis. Angenommen, A sei unimodular und $b \in \mathbb{Z}^m$ ein beliebiger ganzzahliger Vektor. Sei $\bar{x}$ eine Ecke von $\{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$. Wir haben zu zeigen, dass $\bar{x}$ ganzzahlig ist. Da A vollen Zeilenrang hat, gibt es eine Basis $B \subseteq \{1, \ldots, n\}, |B| = m$, so dass $\bar{x}_B = A_B^{-1} b$ und $\bar{x}_N = 0$ ist, wobei $N = \{1, \ldots, n\} \setminus B$ gilt. Da A unimodular ist, folgern wir aus der Cramerschen Regel die Integralität von A_B^{-1} und somit auch von $\bar{x}$.

Umgekehrt sei $\{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ ganzzahlig für alle ganzzahligen Vektoren b, ebenso sei $B \subseteq \{1, \ldots, n\}$ mit A_B regulär. Wir haben zu zeigen, dass $\det A_B = \pm 1$ gilt. Sei $\bar{x}$ die zu B gehörige Ecke, d.h. es gilt $\bar{x}_B = A_B^{-1} b$ und $\bar{x}_N = 0$ mit $N = \{1, \ldots, n\} \setminus B$. Nach unserer Voraussetzung ist $\bar{x}_B = A_B^{-1} b$ ganzzahlig für alle ganzzahligen Vektoren b, also insbesondere für die Einheitsvektoren, also ist A_B^{-1} ganzzahlig. Also sind sowohl $\det A_B$ und $\det A_B^{-1}$ ganze Zahlen, woraus $\det A_B = \pm 1$ folgt. □

Folgerung 2.17 *(Satz von Hoffmann und Kruskal [24])*
Sei A eine ganzzahlige Matrix. Dann ist A genau dann total unimodular, wenn $\{x \mid Ax \leq b, x \geq 0\}$ für alle ganzzahligen Vektoren b ganzzahlig ist.

Beweis. Aus Beobachtung 2.14 (a) wissen wir, dass A genau dann total unimodular ist, wenn $[A \; I]$ unimodular ist. Weiterhin gilt für einen ganzzahligen Vektor b, dass $\{x : Ax \leq b, x \geq 0\}$ genau dann ganzzahlig ist, wenn $\{z : [A \; I]z = b, z \geq 0\}$ ganzzahlig ist (siehe Übung). Die Anwendung von Satz 2.16 auf die Matrix $[A \; I]$ zeigt nun die Behauptung. □

Beobachtung 2.14 (b) in Verbindung mit Korollar 2.17 liefert weitere Charakterisierungen von totaler Unimodularität.

Folgerung 2.18 *Sei A eine ganzzahlige Matrix.*

(a) A ist genau dann total unimodular, wenn $\{x \mid a \leq Ax \leq b, l \leq x \leq u\}$ für alle ganzzahligen Vektoren a, b, l, u ganzzahlig ist.

(b) A ist genau dann total unimodular, wenn $\{x \mid Ax = b,\ 0 \leq x \leq u\}$ für alle ganzzahligen Vektoren b und u ganzzahlig ist.

Zwei Anwendungsbeispiele aus der Kombinatorischen Optimierung

Eine interessante Anwendung von total unimodularen Matrizen ist das Max-Flow-Min-Cut Theorem von [12]. Aus Korollar 2.18 wissen wir, dass eine Matrix A genau dann total unimodular ist, wenn für alle ganzzahligen Vektoren c, b, u das Optimum des linearen Programms $\max\{c^T x \mid Ax = b,\ 0 \leq x \leq u\}$ von einem ganzzahligen Punkt angenommen wird. Da A genau dann total unimodular ist, wenn A^T total unimodular ist, gilt darüber hinaus dieselbe Bedingung für das duale lineare Programm, d.h. $\min\{b^T z + u^T y \mid A^T z + y \geq c,\ y \geq 0\}$ hat eine ganzzahlige Optimallösung.

Folgerung 2.19 *Eine ganzzahlige Matrix A ist genau dann total unimodular, wenn für alle ganzzahligen Vektoren u, b, c beide Seiten der Dualitätsgleichung*

$$\max\{c^T x \mid Ax = b,\ 0 \leq x \leq u\} = \min\{b^T z + u^T y \mid A^T z + y \geq c,\ y \geq 0\}$$

ganzzahlige Lösungen x, y und z haben.

Aus Korollar 2.19 leiten wir das Max-Flow-Min-Cut Theorem ab.

Satz 2.20 *Sei $G = (V, E)$ ein gerichteter Graph mit ganzzahligen Bogenkapazitäten u. Seien $s \neq t$ zwei Knoten in G. Dann ist der Wert eines maximalen Flusses von s nach t gleich dem Wert eines minimalen Schnittes, der s und t trennt.*

Beweis. Sei A die Knoten-Kanten Inzidenzmatrix des gerichteten Graphen $G = (V, E \cup \{(t, s)\})$. Offensichtlich ist ein maximaler (s, t)-Fluss die Lösung des linearen Programms $\max\{x_{ts} :\ Ax = 0,\ 0 \leq x \leq u\}$, wobei $u_{ts} := \infty$ gilt. Aus Korollar 2.19 folgt, dass es ganzzahlige optimale Lösungen $\bar{x}, \bar{y}$ und $\bar{z}$ für die Optimierungsprobleme

$$\max\{x_{ts} :\ Ax = 0,\ 0 \leq x \leq u\} = \min\{u^T y :\ A^T z + y \geq e_{ts},\ y \geq 0\}.$$

gibt. In Worten ausgedrückt: der Wert eines maximalen (s, t)-Fluss, den wir $\bar{x}_{ts}$ nennen wollen, ist gleich dem Wert $u^T \bar{y}$, wobei $\bar{y}$ die folgenden Bedingungen erfüllt:

$$\begin{aligned} \bar{z}_u - \bar{z}_v + \bar{y}_{uv} &\geq 0, \quad \text{falls } uv \neq ts \\ \bar{z}_t - \bar{z}_s + \bar{y}_{ts} &\geq 1. \end{aligned}$$

Aus dem Satz vom schwachen komplementären Schlupf, vgl. die Vorlesung „Einführung in die Optimierung“, und $u_{ts} = \infty$ folgern wir, das $\bar{y}_{ts} = 0$ gilt und damit $\bar{z}_t \geq 1 + \bar{z}_s$. Sei $W := \{w \in V : \bar{z}_w \geq \bar{z}_t\}$. Dann gilt $\bar{y}_{vw} \geq 1$ für alle $vw \in \delta^-(W)$ aufgrund von $z_w \geq z_t > z_v$. Daraus folgern wir

$$u^T \bar{y} \geq \sum_{vw \in \delta^-(W)} u_{vw}.$$

Anders ausgedrückt, der maximale (s,t)-Fluss ist größer oder gleich dem Wert des (s,t)-Schnittes $\delta^-(W)$. Da der maximale Fluss nicht größer als der Wert eines (s,t)-Schnittes sein kann, folgt die Behauptung. □

Ein weiterer Anwendungsbereich total unimodularer Matrizen sind ungerichtete Graphen. Sei G ein ungerichteter Graph und A die Kanten-Knoten Inzidenzmatrix von G. Im Folgenden untersuchen wir Bedingungen, unter denen das Polyeder $P = \{x \mid Ax \leq b,\, x \geq 0\}$ ganzzahlig ist. Dies ist ein interessantes Problem, denn für den Fall $b = \mathbb{1}$ entsprechen die ganzzahligen Lösungen in P stabilen Mengen in G (d.h. einer Teilmenge S der Knoten, so dass jedes Paar von Knoten aus S nicht benachbart ist, in Formeln $E(S) = \emptyset$) bzw. zulässigen Lösungen des Set Packing Problems aus Beispiel 1.6. Das folgende Beispiel zeigt, dass A nicht total unimodular ist, falls G einen ungeraden Kreis enthält, also nicht bipartit ist.

Beispiel 2.21 *Sei G ein Graph und C ein ungerader Kreis in G. Betrachte das lineare Programm $\max\{c^T x \mid 0 \leq x \leq \mathbb{1},\, x_i + x_j \leq 1 \,\forall\, ij \in E\}$ mit $c = \chi^{V(C)}$. Dann gilt:*
(a) $x_i^* = \frac{1}{2}$ für $i \in V(C)$, $x_i^* = 0$ für $i \notin V(C)$, löst das lineare Programm.
(b) x^* ist keine Konvexkombination von Inzidenzvektoren von stabilen Mengen.

Beweis. Übung. □

Satz 2.22 *A ist genau dann total unimodular, wenn G bipartit ist.*

Beweis. Um die Notwendigkeit der Bedingung zu erkennen, beachte man, dass $G = (V, E)$ genau dann bipartit ist, wenn E keinen Kreis ungerader Länge enthält. Angenommen, G ist nicht bipartit, dann enthält G einen Kreis $C \subseteq E$ ungerader Länge. Sei $c = \chi^{V(C)}$. Dann wird $\max\{c^T x : Ax \leq \mathbb{1},\, x \geq 0\}$ von einem Vektor x^* mit $x_i^* = \frac{1}{2}$ angenommen, falls Knoten $i \in V(C)$ und $x_i^* = 0$ ist, für den anderen Fall siehe Beispiel 2.21 (a). Überdies kann x^* nicht eine Konvexkombination von Inzidenzvektoren stabiler Mengen in G sein, betrachte dazu Beispiel 2.21 (b). Also gibt es eine Ecke des Polyeders $\{x \in \mathbb{R}^n : Ax \leq \mathbb{1},\, x \geq 0\}$, die nicht als Konvexkombination von Inzidenzvektoren stabiler Mengen darstellbar ist und folglich nicht ganzzahlig ist. Aus der Betrachtung von Korollar 2.17 folgt, dass A nicht total unimodular ist.
Ist umgekehrt G bipartit, so bezeichnen wir mit V_1 und V_2 die beiden Partitionen der Knotenmengen. Sei $b \in \mathbb{Z}^m$ und $c \in \mathbb{R}^n$ eine Zielfunktion. Wir werden im

Verlauf des Beweises zeigen, dass das lineare Programm $\max\{c^T x : Ax \leq b,\, x \geq 0\}$ eine ganzzahlige Lösung hat. Betrachte die folgende Heuristik:
Berechne eine optimale Lösung x^* des linearen Problems $c^* = \max\{c^T x : Ax \leq b,\, x \geq 0\}$. Sei U eine gleich verteilte Zufallszahl auf dem Intervall $[0,1]$. Für $i = 1, \ldots, n$ definiere

$$z_i := \begin{cases} \lceil x_i^* \rceil & \text{falls } i \in V_1 \text{ und } x_i^* - \lfloor x_i^* \rfloor \geq U, \\ \lceil x_i^* \rceil & \text{falls } i \in V_2 \text{ und } x_i^* - \lfloor x_i^* \rfloor > 1 - U, \\ \lfloor x_i^* \rfloor & \text{sonst.} \end{cases}$$

z ist ein ganzzahliger Punkt mit $z \geq 0$. Betrachte die i-te Zeile von A, die sich zu $e^k + e^l$ mit $kl \in E$ und $k \in V_1$, $l \in V_2$ ergibt.
Gilt $\lfloor x_k^* \rfloor + \lfloor x_l^* \rfloor = b_i$, dann bestimmt die Heuristik den Wert $\lfloor x_k^* \rfloor$ zu z_k und den Wert $\lfloor x_l^* \rfloor$ zu z_l. Wir folgern $z_k + z_l = b_i$.
Gilt $\lfloor x_k^* \rfloor + \lfloor x_l^* \rfloor \leq b_i - 2$, dann folgt $\lceil x_k^* \rceil + \lceil x_l^* \rceil \leq b_i$. In diesem Fall folgern wir $z_k + z_l \leq b_i$.
Andernfalls ist $\lfloor x_k^* \rfloor + \lfloor x_l^* \rfloor = b_i - 1$. Dann ist $x_k^* - \lfloor x_k^* \rfloor + x_l^* - \lfloor x_l^* \rfloor \leq 1$. Ist $x_k^* - \lfloor x_k^* \rfloor \geq U$, dann gilt $x_l^* - \lfloor x_l^* \rfloor < 1 - U$ und umgekehrt. Also werden nicht beide Werte x_k^* und x_l^* aufgerundet, woraus $z_k + z_l \leq b_i$ folgt.
Dies zeigt $Az \leq b$. Da U eine gleichverteilte Zufallsvariable im Intervall $[0,1]$ ist, ist dies $1 - U$ ebenso. Wir erhalten $\text{Prob}(z_k = \lceil x_k^* \rceil) = \text{Prob}(U \leq x_k^* - \lfloor x_k^* \rfloor) = x_k^* - \lfloor x_k^* \rfloor$, falls $k \in V_1$ gilt. Für $l \in V_2$ erhalten wir: $\text{Prob}(z_l = \lceil x_l^* \rceil) = \text{Prob}(1 - U < x_l^* - \lfloor x_l^* \rfloor) = 1 - \text{Prob}(1 - U \geq x_l^* - \lfloor x_l^* \rfloor) = 1 - \text{Prob}(U \leq 1 - (x_l^* - \lfloor x_l^* \rfloor)) = 1 - (1 - (x_l^* - \lfloor x_l^* \rfloor)) = x_l^* - \lfloor x_l^* \rfloor$. Insgesamt haben wir gezeigt:

$$\text{Prob}(z_k = \lceil x_k^* \rceil) = x_k^* - \lfloor x_k^* \rfloor, \text{ für alle } k \in V.$$

Bezeichnen wir mit c^{IP} den Zielfunktionswert einer maximalen stabilen Menge in G bzgl. c mit $c^H = c^T z$ den Zielfunktsionswert der heuristischen Lösung und mit $E(\cdot)$ den Erwartungswertoperator, so erhalten wir

$$\begin{aligned} c^* \geq c^{IP} \geq E(c^H) &= \textstyle\sum_{i \in V} c_i \lfloor x_i^* \rfloor + \sum_{i \in V} c_i \text{Prob}(z_i = \lceil x_i^* \rceil) \\ &= \textstyle\sum_{i \in V} c_i \lfloor x_i^* \rfloor + \sum_{i \in V} c_i (x_i^* - \lfloor x_i^* \rfloor) \\ &= c^*. \end{aligned}$$

Es folgt $c^* = c^{IP}$. Also hat das Polyeder $\{x \in \mathbb{R}^n : Ax \leq b,\, x \geq 0\}$ nur ganzzahlige Werte. Korollar 2.17 impliziert, dass A total unimodular ist. □

Folgerung 2.23 *Betrachte das Zuordnungsproblem aus Beispiel 1.4. Die LP-Relaxierung des binären Programms (die man bekommt, indem man $x_{ij} \in \{0,1\}$ durch $x_{ij} \in [0,1]$ ersetzt) hat immer eine ganzzahlige Optimallösung.*

2.3 Die Hermitesche Normalform

In diesem Abschnitt zeigen wir, dass jede rationale Matrix in eine bestimmte Form, die sogenannte Hermitesche Normalform, gebracht werden kann. Diese Form wird uns helfen, ein ganzzahliges Analogon zum Farkas-Lemma zu beweisen. Dieses Lemma werden wir noch häufiger benötigen, um die Ganzzahligkeit bestimmter Polyeder zu zeigen.

Definition 2.24

(a) *Die folgenden Operationen an einer Matrix heißen* unimodulare (elementare) Spalten- (Zeilen-) Operationen*:*

1. *Vertauschen zweier Spalten (Zeilen).*
2. *Multiplikation einer Spalte (Zeile) mit* -1*.*
3. *Addition eines ganzzahligen Vielfachen einer Spalte (Zeile) zu einer anderen Spalte (Zeile).*

(b) *Eine Matrix* A *mit vollem Zeilenrang ist in* Hermitescher Normalform*, wenn sie die Form* $[B, 0]$ *hat, wobei* B *eine reguläre, nicht-negative, untere Dreiecksmatrix ist und jede Zeile genau einen maximalen Eintrag auf der Hauptdiagonalen hat.*

Satz 2.25 *Jede rationale Matrix mit vollem Zeilenrang kann mit elementaren Spaltenoperationen in Hermitesche Normalform gebracht werden.*

Beweis. Sei A eine rationale Matrix mit vollem Zeilenrang. Wir können annehmen, A sei ganzzahlig (andernfalls skaliere die Matrix mit einem geeigneten Faktor, der am Ende der durchzuführenden Operationen wieder dividiert wird). Wir zeigen zunächst, dass A in die Matrix $[B, 0]$ transformiert werden kann, wobei B eine untere Dreiecksmatrix und $B_{ii} > 0$ für alle i ist. Angenommen, wir haben A bereits in die Form

$$\begin{bmatrix} B & 0 \\ C & D \end{bmatrix},$$

gebracht, wobei B eine untere Dreiecksmatrix mit positiver Diagonale ist. Mittels elementarer Spaltenoperationen bringen wir die erste Zeile von $D \in \mathbb{R}^k$ in eine Form, so dass $D_{11} \geq D_{12} \geq \ldots \geq D_{1k} \geq 0$ gilt und $\sum_{i=1}^{k} D_{1i}$ minimal ist. Da A vollen Zeilenrang hat, folgt $D_{11} > 0$. Wir behaupten, dass $D_{1i} = 0$ für $i = 2, \ldots, k$ gilt. Falls $D_{12} > 0$ ist, subtrahieren wir die zweite Spalte von der ersten und nach eventueller Umordnung der ersten und zweiten Spalte erhalten wir wieder die Elemente der ersten Zeile in nicht steigender Reihenfolge, wobei deren Summe streng streng monoton abnimmt, was ein Widerspruch zur Annahme ist, dass $\sum_{i=1}^{k} D_{1i}$ minimal ist. Nachdem wir diese Schritte n mal durchgeführt haben,

haben wir A nach $[B, 0]$ transformiert, wobei B eine untere Dreiecksmatrix mit positiver Hauptdiagonale ist.

Um Hermitesche Normalform zu erreichen, müssen wir $0 \leq B_{ij} < B_{ii}$ für jedes $i > j$ garantieren können. Dies kann erreicht werden, indem wir ein entsprechendes ganzzahliges Vielfaches von Spalte i zu Spalte j für jedes $i > j$ addieren. Die Reihenfolge, in der diese Operationen durchgeführt werden müssen, ist $(2,1),(3,1),(3,2),(4,1),(4,2),\ldots$. □

Bemerkung 2.26

(a) *Die Operationen, die im Beweis von Satz 2.25 ausgeführt wurden, können durch eine unimodulare Matrix ausgedrückt werden, d.h. es gibt eine unimodulare Matrix U, so dass*

$$[B, 0] = AU$$

und $[B, 0]$ ist in Hermitescher Normalform.

(b) *Für jede rationale Matrix mit vollem Zeilenrang gibt es eine* eindeutige *Hermitesche Normalform und damit eine eindeutige unimodulare Matrix U, siehe [35].*

Satz 2.27 *(Ganzzahliges Analogon des Farkas-Lemma)*
Sei $A \in \mathbb{Q}^{m\times n}$ eine rationale Matrix und $b \in \mathbb{Q}^m$ ein rationaler Vektor. Dann hat genau eines der beiden folgenden Systeme eine Lösung.

$$\begin{aligned} Ax &= b \\ x &\in \mathbb{Z}^n \end{aligned} \qquad\qquad \begin{aligned} y^T A &\in \mathbb{Z}^m \\ y^T b &\notin \mathbb{Z} \\ y &\in \mathbb{Q}^m \end{aligned}$$

Beweis. Beide Gleichungssysteme können nicht gleichzeitig eine Lösung haben, da aus der Integralität von x und $y^T A$ auch diejenige von $y^T b = y^T Ax$ folgt.
Angenommen, $y^T A \in \mathbb{Z}^n, y^T b \notin \mathbb{Z}, y \in \mathbb{Q}^m$ habe keine Lösung, d.h. für alle $y \in \mathbb{Q}^m$, für die $y^T A$ ganzzahlig ist, ist $y^T b$ auch ganzzahlig. Wir müssen zeigen, dass $Ax = b$, $x \in \mathbb{Z}^n$ eine Lösung hat.
Zunächst hat $Ax = b$ eine (möglicherweise gebrochene) Lösung, da es anderweitig rationale Vektoren y mit $y^T A = 0$ und $y^T b \neq 0$ gäbe (vgl. das Eliminationsverfahren nach Gauss). Nach entsprechender Reskalierung gibt es ein y mit $y^T A = 0$ und $y^T b = \frac{1}{2}$, Widerspruch!. Wir können also annehmen, dass A vollen Zeilenrang hat. Der Satz ist invariant bzgl. elementarer Spaltenoperationen. Also können wir nach Satz 2.25 annehmen, dass A in Hermitescher Normalform $A = [B, 0]$ ist. Da $B^{-1}[B, 0] = [I, 0]$ eine ganzzahlige Matrix ist, folgt aus unserer Annahme, dass $B^{-1}b$ ebenso ganzzahlig ist (wähle $y = B^{-1}_{i,\cdot}$ für $i = 1,\ldots,m$). Wegen

$$[B, 0] \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix} = b$$

ist der Vektor $x := \binom{B^{-1}b}{0}$ eine ganzzahlige Lösung von $Ax = b$. □

Beispiel 2.28 *Sei $a \in \mathbb{Z}^n$, $\alpha \in \mathbb{Z}$ und $P_I := conv(\{x \in \mathbb{Z}^n \mid a^T x \leq \alpha\})$. Weiterhin sei $k = \gcd(a_1, \ldots, a_n)$ der größte gemeinsame Teiler der Komponenten von a. Dann folgt aus Satz 2.27, dass*

$$P_I = \{x \in \mathbb{R}^n \mid \sum_{i=1}^{n} \tfrac{a_i}{k} x_i \leq \lfloor \tfrac{\alpha}{k} \rfloor\}.$$

Beweis. Übung. □

2.4 Totale Duale Integralität

In Abschnitt 2.2 haben wir gesehen, dass totale Unimodularität einer ganzzahligen Matrix A sicherstellt, dass für jede beliebige rechte Seite b das Polyeder $P_I = \text{conv}\{x \in \mathbb{Z}^n \mid Ax \leq b\}$ durch die Originalungleichungen $Ax \leq b$ vollständig beschrieben wird. Legen wir uns dagegen auf eine bestimmte rechte Seite b fest, dann ist TDI (total dual integrality) das richtige Konzept um Ganzzahligkeit eines Polyeders zu garantieren.

Definition 2.29 *Sei $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$. Das System von Ungleichungen $Ax \leq b$ heißt* total dual integral (TDI)*, falls es für jeden ganzzahligen Vektor $c \in \mathbb{Z}^m$, für den $\min\{b^T y \mid A^T y = c,\ y \geq 0\}$ endlich ist, einen ganzzahligen Vektor $y^* \in \mathbb{Z}^m$ mit $A^T y^* = c$, $y^* \geq 0$ gibt, so dass $b^T y^* = \min\{b^T y \mid A^T y = c, y \geq 0\}$ gilt.*

Die TDI-Eigenschaft eines Systems $Ax \leq b$ hat eine geometrische Bedeutung: Sei c ein ganzzahliger Vektor, der in dem von den Zeilen von A aufgespannten Kegel liegt, d.h es gibt $y \geq 0$ mit $A^T y = c$. Unter allen Möglichkeiten c als konische Kombination der Zeilen von A zu schreiben sei S die Menge der kürzesten (bzgl. b) konischen Kombination, d.h.

$$S = \{y \geq 0 \mid A^T y = c, \text{ so dass } b^T y \text{ minimal ist}\}.$$

Dann ist $Ax \leq b$ TDI, falls es einen ganzzahligen Vektor in S gibt. Dies bedeutet mit anderen Worten also, dass unter allen kürzesten Möglichkeiten, c als konische Kombination der Zeilen von A zu schreiben, eine dabei ist, die ganzzahlig ist. Wir werden sehen, dass dies in engem Zusammenhang mit Hilbert-Basen steht, die in der ganzzahligen Programmierung eine wichtige Rolle spielen.
Aus der Definition der TDI-Eigenschaft folgt direkt, dass $Ax \leq b$ TDI ist, falls die Matrix A total unimodular ist.
Beachte auch, dass TDI eine Eigenschaft eines Ungleichungssystems ist und nicht eine Eigenschaft des Polyeders $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. Wir werden sehen, dass es viele Möglichkeiten geben kann P zu beschreiben, aber es gibt nur eine, die

auch die TDI-Eigenschaft besitzt. Dazu kann es notwendig sein, viele redundante Ungleichungen zur Ausgangsformulierung hinzufügen zu müssen.
Der folgende Satz gibt uns den Zusammenhang zwischen der TDI-Eigenschaft eines Ungleichungssystems $Ax \leq b$ und der Ganzzahligkeit des zugehörigen Polyeders $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$.

Satz 2.30 *Ist $Ax \leq b$ TDI und b ganzzahlig, dann ist $\{x \in \mathbb{R}^n \mid Ax \leq b\}$ ganzzahlig.*

Beweis. Sei $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ und $\emptyset \neq F \neq P$ eine minimale Seitenfläche von P. Wir setzen $k := |\text{eq}(F)|$. Da F minimal ist, folgt $F = \{x \in P : A_{\text{eq}(F)}x = b_{\text{eq}(F)}\} = \{x \in \mathbb{R}^n : A_{\text{eq}(F)}x = b_{\text{eq}(F)}\}$. Wir müssen zeigen, dass F ganzzahlige Punkte enthält.
Enthält F keine ganzzahligen Punkte, so existiert ein $y \in \mathbb{Q}^k$ mit $c := y^T A_{\text{eq}(F)} \in \mathbb{Z}^n$ und $\gamma := y^T b_{\text{eq}(F)} \notin \mathbb{Z}$, siehe Satz 2.27. Wir können annehmen, dass y nichtnegativ ist (andernfalls wähle $s \in \mathbb{Z}^k_+$, groß genug, so dass $y + s \geq 0$, $(y+s)^T A \in \mathbb{Z}^n$ und $(y+s)^T b_{\text{eq}(F)} \notin \mathbb{Z}$. Wir behaupten, dass $\max\{c^T x : Ax \leq b\}$ existiert und von jedem $\hat{x} \in F$ angenommen wird. Die Behauptung ist korrekt, da für alle $x \in P$ und $\hat{x} \in F$ gilt

$$c^T x = y^T A_{\text{eq}(F)} x \;\leq\; y^T b_{\text{eq}(F)} = y^T A_{\text{eq}(F)} \hat{x} = c^T \hat{x}.$$

Aufrund des Dualitätssatzes der Linearen Optimierung, vgl. die Vorlesung „Einführung in die Optimierung“, und der Tatsache, dass $Ax \leq b$ TDI ist, folgern wir, dass der Wert $\min\{b^T y : A^T y = c, y \geq 0\}$ existiert und für einen ganzzahligen Vektor angenommen wird. Da b ganzzahlig ist, ist der Wert $\min\{b^T y : A^T y = z, y \geq 0\}$ eine ganze Zahl. Diese Zahl muss mit dem Wert $\max\{c^T x : Ax \leq b\} = \gamma \notin \mathbb{Z}$ übereinstimmen, ein Widerspruch. Also enthält F ganzzahlige Punkte. □

Wir haben bereits die geometrische Interpretation von TDI betrachtet und den Zusammenhang zu Hilbert-Basen angedeutet. Dies wollen wir nun präzisieren.

Definition 2.31 *Sei $C \subseteq \mathbb{R}^n$ ein rationaler polyedrischer Kegel. Eine endliche Teilmenge $H = \{h^1, \ldots, h^t\} \subset C$ heißt* Hilbert-Basis *von C, falls sich jeder ganzzahlige Punkt $z \in C \cap \mathbb{Z}^n$ als nicht-negative ganzzahlige Linearkombination von Elementen aus H darstellen lässt, d.h.*

$$z = \sum_{i=1}^{t} \lambda_i h^i$$

mit $\lambda_1, \ldots, \lambda_t \in \mathbb{N}_0$. Eine Hilbert-Basis heißt ganzzahlig, *falls $H \subseteq \mathbb{Z}^n$ ist.*

Beispiel 2.32 *Betrachte $C = \text{cone}(\{\binom{1}{3}, \binom{2}{1}\})$. Dann ist $H = \{\binom{1}{3}, \binom{1}{2}, \binom{1}{1}, \binom{2}{1}\}$ eine Hilbert-Basis von C, vgl. Abbildung 2.1.*

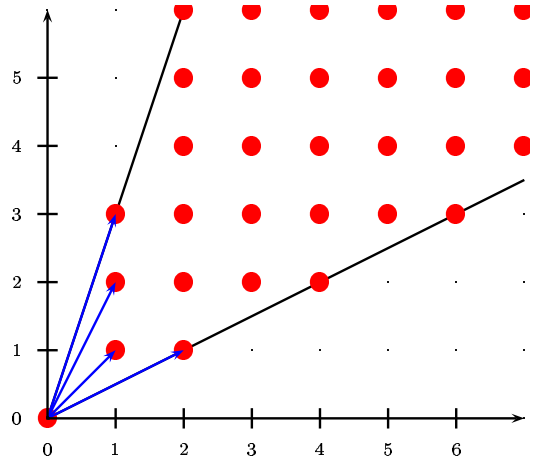

Abbildung 2.1: Hilbert-Basis eines polyedrischen Kegels

Bemerkung 2.33

(a) Für jeden rationalen polyedrischen Kegel existiert eine ganzzahlige Hilbert-Basis.

(b) Falls C spitz ist, ist die ganzzahlige Hilbert-Basis eindeutig bestimmt.

Beweis. Zum Beweis siehe beispielsweise Schrijver [35], Kapitel 16.4. □

Satz 2.34 *Sei $A \in \mathbb{Q}^{m \times n}$ und $b \in \mathbb{Q}^m$. Das Ungleichungssystem $Ax \leq b$ hat genau dann die TDI-Eigenschaft, wenn für jede Seitenfläche F von $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ die Zeilen von $A_{eq(F)\cdot}$ eine Hilbert-Basis des Kegels $C(A_{eq(F)\cdot})$ bilden.*

Um den Beweis von Satz 2.34 nachzuvollziehen, ist folgende geometrische Interpretation hilfreich: Sei c ein ganzzahliger Vektor, der in dem von den Zeilen von A aufgespannten Kegel liegt, so dass der Wert $\min\{b^T y \mid A^T y = c,\, y \geq 0\}$ existiert. Dann wissen wir, dass c das Optimum an einer Seitenfläche von P annimmt. Aus der Theorie der Linearen Programmierung (vgl. den Satz vom komplementären Schlupf aus der Vorlesung „Einführung in die Optimierung“) wissen wir, dass wenn c das Optimum an der Seitenfläche F annimmt, dann liegt c in dem von den Zeilen von $A_{\mathrm{eq}(F)\cdot}$ aufgespannten Kegel. Damit existiert der Wert $\min\{b^T y \mid A^T y = c,\, y \geq 0\}$. Die Bedingung, dass die von F induzierten Zeilen eine Hilbert-Basis bilden, ist äquivalent dazu, dass c als nicht-negative ganzzahlige Kombination aus von F induzierten Zeilen dargestellt werden kann. In Formeln bedeutet dies, es gibt einen ganzzahligen Vektor $y^* \in \mathbb{Z}^m$, $A^T y^* = c$, $y^* \geq 0$ mit $b^T y^* = \min\{b^T y \mid A^T y = c, y \geq 0\}$. Dies ist die Definition der TDI-Eigenschaft eines Ungleichungssystems $Ax \leq b$.

Beweis. Sei $Ax \leq b$ TDI. Weiter sei $F \neq \emptyset$ eine Seitenfläche von P und $c \in C(A_{\text{eq}(F)}) \cap \mathbb{Z}^n$. Wir müssen zeigen, dass c als eine nichtnegative ganzzahlige Kombination der Zeilenvektoren von $A_{\text{eq}(F)}$ dargestellt werden kann. Wir wissen, dass c eine nichtnegative Linearkombination der Zeilenvektoren von $A_{\text{eq}(F)}$ ist, d.h. es gibt ein $\hat{y} \geq 0$ mit $\hat{y}_i = 0$ für alle $i \notin \text{eq}(F)$ und $c = \hat{y}^T A$.
Wir behaupten nun, dass $\max\{c^T x : Ax \leq b\}$ existiert und für jedes $\hat{x} \in F$ angenommen wird. Dies folgt, da für alle $x \in P$ und $\hat{x} \in F$ die Aussage

$$c^T x = \hat{y}^T Ax \leq \hat{y}^T b = \hat{y}^T A\hat{x} = c^T \hat{x}$$

gilt. Aus dem Dualitätssatz der Linearen Optimierung, vgl. die Vorlesung „Einführung in die Optimierung“, schließen wir, dass der Wert $\min\{b^T y : A^T y = c,\, y \geq 0\}$ existiert. Verwenden wir, dass $Ax \leq b$ TDI ist, so erhalten wir

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

und dass es einen ganzzahligen Vektor y^* gibt, der $\min\{b^T y : A^T y = c,\, y \geq 0\}$ annimmt. Da jedes $\hat{x} \in F$ eine optimale Lösung des Programms $\max\{c^T x : Ax \leq b\}$ ist, gibt es $\hat{x}$, so dass $A^T_{i,\cdot}\hat{x} = b_i$ für alle $i \in \text{eq}(F)$ und $A^T_{i,\cdot}\hat{x} < b_i$ für alle $i \notin \text{eq}(F)$ gilt. Aus dem Satz vom schwachem komplementären Schlupf, vgl. die Vorlesung „Einführung in die Optimierung“, folgt $y^*_i = 0$ für alle $i \notin \text{eq}(F)$. Damit ist der erste Teil des Beweises vollendet, da y^* ganzzahlig, $y^*_i = 0$ für alle $i \notin \text{eq}(F)$, $y^*_i \geq 0$ für alle $i \in \text{eq}(F)$ und c eine nichtnegative ganzzahlige Kombination der Vektoren von $A_{\text{eq}(F)}$ ist.
Für die Umkehrung sei $c \in \mathbb{Z}^n$, so dass $\gamma := \min\{b^T y : A^T y = c,\, y \geq 0\}$ existiert. Aus dem Dualitätssatz der Linearen Optimierung, vgl. die Vorlesung „Einführung in die Optimierung“, wissen wir, dass der Wert $\gamma := \max\{c^T x : Ax \leq b\}$ existiert. Sei $F = \{x \in \mathbb{R}^n : Ax \leq b, c^T x = \gamma\}$ eine Seitenfläche von P, die den Optimalwert der Funktion c annimmt. Wir bemerken, dass $c \in C(A_{\text{eq}(F)})$ aufgrund des Satzes vom schwachem komplementären Schlupf, vgl. die Vorlesung „Einführung in die Optimierung“, gilt und es gibt einen Vektor $y^* \geq 0$ mit $y^*_i = 0$ für alle $i \notin \text{eq}(F)$ und es ist $c = \sum_{i \in \text{eq}(F)} A_{i\cdot} y^*_i$. Da die Menge $A_{\text{eq}(F)}$ eine Hilbert-Basis von $C(A^T_{\text{eq}(F)})$ ist, gibt es einen nichtnegativen ganzzahligen Vektor $\tilde{y}$, so dass $c = \sum_{i \in \text{eq}(F)} A_{i,\cdot} \tilde{y}_i$. Setzen wir $y_i := 0$ für alle $i \notin \text{eq}(F)$ und $y_i := \tilde{y}_i$ für alle $i \in \text{eq}(F)$, so haben wir den ganzzahligen Vektor y gefunden, so dass $A^T y = c$ und $y \geq 0$ ist. Weiter gilt für $x \in F$

$$b^T y = \sum_{i \in \text{eq}(F)} y_i b_i = \sum_{i \in \text{eq}(F)} y_i A^T_{i\cdot} x = c^T x = \gamma.$$

Also ist $Ax \leq b$ total dual integral. □

Als Anwendung von Satz 2.34 zeigen wir, dass TDI eines Ungleichungssystems erhalten bleibt, wenn wir uns auf Teilsysteme, die Seitenflächen des mit dem Originalsystem assoziierten Polyeders induzieren, beschränken.

Folgerung 2.35 *Sei $A \in \mathbb{Q}^{m\times,n}, b \in \mathbb{Q}^m, c \in \mathbb{Q}^n, d \in \mathbb{Q}$. Hat das Ungleichungssystem $Ax \leq b, c^T x \leq d$ die Eigenschaft TDI zu sein, dann ist auch das System $Ax \leq b$, $c^T x \leq d$, $-c^T x \leq -d$ TDI.*

Beweis. Sei F eine Seitenfläche des Polyeders $\operatorname{conv}\{x \in \mathbb{R}^n : Ax \leq b, c^T x \leq d, -c^T x \leq -d\}$. Dann ist F ebenso eine Seitenfläche des Polyeders $\operatorname{conv}\{x \in \mathbb{R}^n : Ax \leq b, c^T x \leq d\}$. Da $Ax \leq b, c^T x \leq d$ TDI ist, folgern wir mit Satz 2.34, dass $A^T_{\text{eq}(F)}, c$ eine Hilbert-Basis des Kegels $C(A^T_{\text{eq}(F)}, c)$ ist. Sei $z \in C(A^T_{\text{eq}(F)}, c, -c) \cap \mathbb{Z}^n$. Dann ist z von der Form

$$z = \sum_{i\in\text{eq}(F)} \lambda_i A_{i\cdot} + \mu c - \sigma c$$

mit $\lambda_i \geq 0$, $i \in \text{eq}(F)$, $\mu \geq 0$, $\sigma \geq 0$. Dies ist äquivalent zu $z + \sigma c \in C(A^T_{\text{eq}(F)}, c)$. Sei $\sigma' \in \mathbb{N}$, $\sigma' \geq \sigma$ so gewählt, dass $\sigma' c \in \mathbb{Z}$ ist. Es folgt $z + \sigma' c \in C(A^T_{\text{eq}(F)}, c)$. Da $A^T_{\text{eq}(F)}, c$ eine Hilbert-Basis von $C(A^T_{\text{eq}(F)}, c)$ ist, können wir $z + \sigma' c$ auch schreiben als

$$z + \sigma' c = \sum_{i\in\text{eq}(F)} \lambda'_i A_{i\cdot} + \mu' c$$

wobei $\lambda'_i \geq 0$ ganzzahlig ist für alle $i \in \text{eq}(F)$ und ganzzahliges $\mu' \geq 0$. Es folgt, dass

$$z = \sum_{i\in\text{eq}(F)} \lambda'_i A_{i\cdot} + \mu' c - \sigma' c,$$

d.h. z kann als nichtnegative ganzzahlige Kombination der Erzeugenden des Kegels $C(A^T_{\text{eq}(F)}, c, -c)$ ausgedrückt werden. Daraus folgt, dass $A^T_{\text{eq}(F)}, c, -c$ eine Hilbert-Basis von $C(A^T_{\text{eq}(F)}, c, -c)$ ist. Aus Satz 2.34 folgt nun die Behauptung. □

Das folgende Theorem gibt eine weitere Charakterisierung von TDI über Hilbert-Basen. Diese ist hilfreich, wenn man algorithmisch überprüfen möchte, ob ein Ungleichungssystem TDI ist.

Satz 2.36 *Sei $A \in \mathbb{Q}^{m\times n}$ und $b \in \mathbb{Q}^m$ rational, so dass $P = \{x \in \mathbb{R}^n \mid Ax \leq b\} \neq \emptyset$ gilt. Das System $Ax \leq b$ ist genau dann TDI, wenn*

(1) die Zeilen von A eine Hilbert-Basis des Kegels, der von den Zeilen von A aufgespannt wird, bilden, und

(2) für jede Teilmenge $S \subseteq \{1, \ldots, m\}$ das lineare Programm

$$\min\{b^T y \mid A^T y = \sum_{i\in S} A_{i\cdot},\ y \geq 0\}$$

eine ganzzahlige Optimallösung hat.

Beweis. Ist $Ax \leq b$ TDI, dann bilden die Zeilen von A eine Hilbert-Basis. Ist dies nicht der Fall, dann gibt es ein $z \in C(A_{1\cdot}, \ldots, A_{m\cdot}) \cap \mathbb{Z}^n$, das nicht als nichtnegative ganzzahlige Kombination von $A_{1\cdot}, \ldots, A_{m\cdot}$ geschrieben werden kann. Da $P \neq \emptyset$, existiert der Wert $\min\{b^T y : A^T y = z, y \geq 0\}$ und die optimale Lösung wird nicht durch einen ganzzahligen Vektor angenommen, ein Widerspruch zur Definition eines TDI-Systems.

Bedingung (2) folgt direkt aus der Definition eines TDI-Systems. Beachte, dass, falls $c := \sum_{i \in S} A_{i\cdot}$ nicht ganzzahlig ist, man ein geeigneten Skalar λ finden kann, so dass $A^T y = c$ eine ganzzahlige Lösung hat genau dann, wenn $\lambda A^T y = \lambda c$ eine ganzzahlige Lösung hat.

Für die umgekehrte Richtung, nehmen wir an, dass die Zeilen von A eine Hilbert-Basis bilden und dass für jede Menge $S \subseteq \{1, \ldots, m\}$ das lineare Optimierungsproblem

$$\min\{b^T y : A^T y = \sum_{i \in S} A_{i\cdot}, y \geq 0\}$$

eine ganzzahlige optimale Lösung hat. Wir müssen zeigen, dass für jedes ganzzahlige $c \in \mathbb{Z}^n$, für das der Wert $\gamma = \min\{b^T y : A^T y = c, y \geq 0\}$ existiert, dieser von einer ganzzahligen Lösung angenommen wird. Da die Zeilen von A eine Hilbert-Basis bilden, wissen wir, dass es einen ganzzahligen Vektor y gibt, so dass $A^T y = c$, $y \geq 0$ ist. Sei y^* eine ganzzahlige Lösung von $A^T y^* = c$, $y^* \geq 0$, wobei $b^T y^*$ so klein als möglich ist. Definiere S als den Support von y^*, d.h. $S = \{i \in \{1, \ldots, m\} : y_i^* > 0\}$.

Wir zeigen als Zwischenschritt, dass $\sum_{i \in S} b_i$ der optimale Wert des Hilfsproblems $\min\{b^T y : A^T y = \sum_{i \in S} A_{i\cdot}, y \geq 0\}$ ist. Nach unserer Annahme gibt es eine ganzzahlige optimale Lösung u^* von $\min\{b^T y : A^T y = \sum_{i \in S} A_{i\cdot}, y \geq 0\}$. Ist $(u^*)^T b < \sum_{i \in S} b_i = (\sum_{i \in S} e^i)^T b$, dann ist v definiert durch $v_i := u_i^*$ für alle $i \notin S$ und $v_i := u_i^* - 1$ für alle $i \in S$ ganzzahlig und erfüllt $v \geq -1$. Dann ist $y^* + v$ größer oder gleich 0 und ganzzahlig, ebenso $A^T(y^* + v) = A^T(y^*) + A^T u^* - A^T \sum_{i \in S} e^i = c$ und $b^T(y^* + v) < b^T(y^*)$, was ein Widerspruch zur Wahl von y^* ist.

Wir folgern, dass $\sum_{i \in S} b_i$ das Optimum des Hilfsproblems $\min\{b^T y : A^T y = \sum_{i \in S} A_{i\cdot}, y \geq 0\}$ ist und dass $\sum_{i \in S} e^i$ eine optimale Lösung für dieses Problem ist, die den Zielfunktsionswert $\sum_{i \in S} b_i$ annimmt. Aus dem Satz vom komplementären Schlupf folgt die Existenz eines Vektors $x^* \in P$, so dass $A_{i\cdot}^T x^* = b_i$ für alle $i \in S$, $A_{i\cdot}^T x^* \leq b_i$ für alle $i \notin S$ ist und dass x^* die Zielfunktion $\sum_{i \in S} A_{i\cdot}$ maximiert. Damit haben wir ein Paar x^*, y^* von primal, dual zulässigen Lösungen mit

$$c^T x^* = (y^*)^T A x^* = \sum_{i \in S} y_i^* A_{i\cdot}^T x^* = \sum_{i \in S} y_i^* b_i = b^T y^*.$$

Es folgt, dass der ganzzahlige Vektor y^* das Problem $\min\{b^T y : A^T y = c, y \geq 0\}$ löst. □

Der folgende Satz zeigt, dass es zu jedem rationalen Polyeder P ein Ungleichungssystem mit der TDI-Eigenschaft gibt. Man erhält es, indem man zu jeder minimalen Seitenfläche F von P eine Hilbert-Basis des Kegels berechnet, der durch die F induzierenden Ungleichungen aufgespannt wird.

Satz 2.37 *Jedes rationale Polyeder P kann durch ein TDI-System der Form $Ax \leq b$, wobei A ganzzahlig ist, beschrieben werden.*

Beweis. Sei $P = P(D, d)$ mit $D \in \mathbb{Q}^{m \times n}$, $d \in \mathbb{Q}^m$ und $Dx \leq d$ nicht redundant. Seien $F_1, \ldots, F_t$ alle (minimalen) Seitenflächen von P. Sei $\mathcal{H}_i \subseteq \mathbb{Z}^n$ eine minimale ganzzahlige Hilbert-Basis des Kegels $C(D^T_{\text{eq}(F_i)})$. Definiere A als die Matrix, deren Zeilen zu allen Vektoren in $\bigcup_{i=1}^t \mathcal{H}_i$ gehören. Betrachte den k-ten Zeilenvektor $A_{k\cdot}$ von A. Dann ist $A_{k\cdot}$ ein Element einer ganzzahligen Hilbert-Basis, wir nennen diese $\mathcal{H}_i$. Also ist A eine ganzzahlige Matrix. Wir definieren den k-ten Koeffizienten des Vektors b zu $\max\{(A_{k\cdot})^T x : x \in P\}$. Beachte, dass dieses Maximum existiert, da für $A_{k\cdot} \in \mathcal{H}_i$ gilt

$$b_k = \max\{A^T_{k\cdot} x : x \in P\} = A^T_{k\cdot}\hat{x} \text{ für alle } \hat{x} \in F_i, \tag{2.2}$$

wegen $A_{k\cdot} \in C(D^T_{\text{eq}(F_i)})$.
Wir behaupten $P = \{x \in \mathbb{R}^n : Ax \leq b\}$.
(1) Sicherlich gilt

$$P \subseteq \{x \in \mathbb{R}^n : Ax \leq b\},$$

da für jedes $y \in P$ und Zeilenindex k der Matrix A gilt: $A^T_{k\cdot} y \leq \max\{A^T_{k\cdot} x : x \in P\} = b_k$.
(2) Ist umgekehrt $y \notin P$, dann gibt es einen Zeilenindex $l \in \{1, \ldots, m\}$, so dass $D_{l\cdot} y > d_l$ ist. Sei $i \in \{1, \ldots, t\}$, so dass $l \in \text{eq}(F_i)$. Der Index i ist wohldefiniert, da D nicht redundant ist.

Sei $\delta \geq 0$ ein Skalar, so dass $\delta D_{l\cdot} \in \mathbb{Z}^n$. Es gibt nichtnegative ganzzahlige Vielfache $\delta_1, \ldots, \delta_s$ für die Elemente in $\{a^1, \ldots, a^s\} = \mathcal{H}_i$, so dass

$$\delta D_{l\cdot} = \sum_{w=1}^{s} \delta_w a^w.$$

Unter Berücksichtigung von (2.2) erhalten wir für $\hat{x} \in F_i$

$$\sum_{w=1}^{s} \delta_w (a^w)^T y = \delta D_{l\cdot} y > \delta d_l = \delta D_{l\cdot}\hat{x} = \sum_{w=1}^{s} \delta_w (a^w)^T \hat{x} = \sum_{w=1}^{s} \delta_w b_w,$$

wobei b_w diejenige Komponente der rechten Seite b ist, die zur Zeile a^w von A gehört. Also gibt es ein $\hat{w} \in \{1, \ldots, s\}$, so dass $(a^{\hat{w}})^T y > b_{\hat{w}}$. Das vollendet den Beweis, dass $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ ist.
Das System $Ax \leq b$ ist TDI, denn ist F_i eine Seitenfläche von P und $\{a^1, \ldots, a^s\} = \mathcal{H}_i$, so ist $(a^w)^T x = b_w$ für alle $x \in F_i$ und $w \in \{1, \ldots, s\}$, siehe

(2.2). Zusätzlich ist $\{a^1, \ldots, a^s\}$ eine ganzzahlige Hilbert-Basis von $C(A^T_{\text{eq}(F_i)}) = C(D^T_{\text{eq}(F_i)})$. Unter Anwendung von Satz 2.34 folgt die Behauptung. □

In der Tat gilt sogar, dass P genau dann ganzzahlig ist, wenn b ganzzahlig gewählt werden kann. Die eine Richtung erhält man aus dem Beweis von Satz 2.37, denn P ist genau dann ganzzahlig, wenn jede minimale Seitenfläche einen ganzzahligen Punkt enthält. Konstruieren wir unser TDI-System wie im Beweis von Satz 2.37, so erfüllt jeder ganzzahlige Punkt z einer Seitenfläche einige der Ungleichungen von $Ax \leq b$ mit Gleichheit. Aufgrund von (2.2) müssen die zugehörigen Komponenten von b ganzzahlig sein. Umgekehrt impliziert Satz 2.30 im Falle der Ganzzahligkeit von b die Ganzzahligkeit von P. Damit gilt

Folgerung 2.39 *Ein rationales Polyeder P ist genau dann ganzzahlig, wenn es ein TDI- System der Form $Ax \leq b$ (mit A und b ganzzahlig) gibt, das P beschreibt.*

Anwendungsbeispiele

Es gibt viele Beispiele für TDI-Systeme, von denen hier exemplarisch zwei genannt seien.
Eine 0/1 Matrix A heißt balanciert, falls sie keine ungerade quadratische Untermatrix mit zwei Einsen pro Zeile und Spalte enthält. [13] haben gezeigt, dass das System $Ax \leq \mathbb{1}$, $x \geq 0$ die TDI-Eigenschaft besitzt, falls A balanciert ist.
Häufig werden TDI Systeme auch dazu verwendet, diskrete Min-Max Resultate zu beweisen, vgl. das Max-Flow-Min-Cut Theorem 2.20. Dazu betrachten wir folgendes Beispiel:
Sei $G = (V, E)$ ein gerichteter Graph. Sei $|E| = m$ und wähle einen bestimmten Knoten $r \in V$. Eine r-Arboreszenz ist eine Teilmenge E' von E mit $|E'| = |V| - 1$ Bögen, so dass es für jeden Knoten $v \neq r$ genau einen Bogen in E' gibt, dessen Endknoten v ist. In Formeln,

$$|\delta^-(v) \cap E'| = 1 \text{ für alle } v \in V \setminus \{r\}.$$

Ein r-Schnitt ist eine Teilmenge der Bögen der Form $\delta^-(U)$ mit $\emptyset \neq U \subset V \setminus \{r\}$. Sei C die Menge aller r-Schnitte von G und bezeichne A die $\pm 1/0$ Matrix, deren Zeilen die Inzidenzvektoren aller r-Schnitte in C repräsentieren. [10] haben gezeigt, dass das System

$$\{x \in \mathbb{R}^m \mid Ax \geq \mathbb{1},\, x \geq 0\} \tag{2.3}$$

TDI ist.
Mit diesem Resultat kann man einfach Fulkersons Satz von der optimalen Arboreszenz [15] beweisen. Sei $c \in \mathbb{N}_0^m$ ein nicht-negativer ganzzahliger Vektor. Aufgrund von (2.3) und der LP-Dualität erhalten wir, dass

$$\min\{c^T x \mid Ax \geq \mathbb{1},\, x \geq 0\} = \max\{\mathbb{1}^T y \mid A^T y \leq c,\, y \geq 0\}$$

und beide Optima von ganzzahligen Punkten x^* und y^* angenommen werden, siehe Satz 2.30. Das heißt, die minimale (bzgl. c) r-Arboreszenz ist gleich der maximalen Anzahl von r-Schnitten, wobei kein Bogen $uv \in E$ in mehr als c_{uv} r-Schnitten enthalten ist.

Kapitel 3

Relaxierungen

Betrachten wir ein allgemeines gemischt-ganzzahliges Programm wie in Definition 1.1 eingeführt.

$$\begin{aligned} \min \quad & c^T x \\ & Ax \leq b \\ & x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p, \end{aligned} \tag{3.1}$$

wobei $A \in \mathbb{R}^{m \times n}, c \in \mathbb{R}^n, b \in \mathbb{R}^m$ und $p = \{0, \ldots, n\}$.
Im vorigen Kapitel haben wir untersucht, unter welchen Bedingungen wir ganzzahlige Lösungen bekommen, ohne dass wir dies explizit fordern mussten und $x \in \mathbb{Z}^{n-p}$ durch $x \in \mathbb{R}^{n-p}$ setzen konnten. In diesem Kapitel behandeln wir Methoden, die versuchen mit dem allgemeinen ($\mathcal{NP}$-schweren) Fall umzugehen. Die Grundidee aller Methoden ist, einen Teil, der das Problem schwierig macht, los zu werden. Sie unterscheiden sich in der Art und Weise, wie der gelöschte Teil in das Problem zurückgeführt wird. In Kapitel 3.1 betrachten wir LP-Relaxierungen. Hier vernachlässigen wir die Ganzzahligkeitsbedingungen und versuchen diese durch Hinzufügen von Schnittebenen zu erreichen. In Kapitel 3.2 löschen wir einen Teil der Nebenbedingungen und transferieren diese in die Zielfunktion versehen mit einem Strafparamater für Nichteinhaltung der Bedingungen. In Kapitel 3.3 diskutieren wir Dekompositionsmethoden, u.a. Dantzig-Wolfe und Benders' Dekomposition. Diese Methoden löschen ebenfalls einen Teil der Nebenbedingungsmatrix, reformulieren den gelöschten Teil und fügen den reformulierten Teil wieder in das Gesamtproblem ein.

3.1 LP-Relaxierungen

Relaxieren wir die Ganzzahligkeitsbedingungen in (3.1) so erhalten wir die sog. LP-Relaxierung von (3.1):

$$\begin{aligned} \min \quad & c^T x \\ & Ax \leq b \\ & x \in \mathbb{R}^n. \end{aligned} \tag{3.2}$$

Zur Lösung linearer Programme sind polynomiale und effiziente Methoden bekannt, wie wir sie ausführlich in der Vorlesung „Einführung in die Optimierung“ diskutiert haben. Falls die optimale Lösung x^* der LP-Relaxierung ganzzahlig ist, haben wir (3.1) gelöst. Andernfalls muss es eine Ungleichung geben (auch Schnittebene genannt) die x^* von $P_I = \text{conv}\left(\{x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p \mid Ax \leq b\}\right)$ trennt. Das Problem, eine solche Ungleichung zu finden, nennt man das Separierungsproblem, siehe Vorlesung „Einführung in die Optimierung“. Finden wir eine solche Ungleichung, so verbessern wir die LP-Relaxierung, indem wir die Ungleichung dem LP hinzufügen und fahren fort. Dieser Gesamtprozess wird auch als Schnittebenenverfahren bezeichnet. Da wir wissen, dass SEP und OPT äquivalent sind, vgl. die Vorlesung „Einführung in die Optimierung“, können wir nicht erwarten, auf diese Weise immer eine Optimallösung zu finden. Falls wir das nicht tun, d.h. die optimale LP Lösung ist gebrochen und wir finden keine weiteren Ungleichungen, betten wir das Verfahren in ein Enumerationsschema ein, siehe dazu Kapitel 4. Wir hoffen aber, durch das Schnittebenenverfahren die Ausgangsformulierung so zu verbessern, dass der anschließende Enumerationsaufwand gering bleibt.
Der Schlüssel zum Erfolg dieser Methode ist also, gute Ungleichungen für das zugrundeliegende Polyeder zu kennen/finden. Wir diskutieren zunächst Wege, Ungleichungen zu generieren, die unabhängig von der zugrundeliegenden Problemstruktur sind und schauen dann auf MIPs mit spezieller lokaler Struktur.

3.1.1 Allgemeine Schnittebenen

In diesem Kapitel behandeln wir eine Klasse von Ungleichungen, die gültig für P_I ist und die unabhängig von jeder Problemstruktur angewandt werden kann ([19], [6]). Wir werden sehen, dass diese Klasse das Potential hat, P_I vollständig zu beschreiben. Es gibt zwei Zugänge für diese Ungleichungen, einen geometrischen und einen algorithmischen. Wir beginnen mit dem geometrischen Ansatz und beschränken uns zunächst auf rein ganzzahlige Probleme.

Chvátals geometrischer Zugang

Betrachten wir ein rationales Polyeder

$$P := \{x \in \mathbb{R}^n : \; Ax \leq b\}$$

mit $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$. Wir wollen eine lineare Beschreibung von

$$P_I := \text{conv}\{x \in \mathbb{Z}^n : Ax \leq b\}.$$

finden. Nach Beobachtung 2.12 ist $P = P_I$ genau dann, wenn jede (minimale) Seitenfläche von P ganzzahlige Punkte enthält. Dies wiederum ist äquivalent dazu, dass jede Stützhyperebene ganzzahlige Punkte enthält.

Lemma 3.2 *Sei P ein Polyeder. Dann enthält jede minimale Seitenfläche von P ganzzahlige Punkte genau dann, wenn jede Stützhyperebene ganzzahlige Punkte enthält.*

Beweis. Übung. □

Die Idee der Methode, die wir nun diskutieren, ist, sich jede Stützhyperebene von P anzuschauen und sie solange näher an P_I zu schieben, bis sie einen ganzzahligen Punkt enthält.
Sei $\{x \in \mathbb{R}^n : c^T x = \delta\}$ eine Stützhyperebene P mit $P \subseteq \{x \in \mathbb{R}^n : c^T x \leq \delta\}$ und c ganzzahlig. Offensichtlich gilt

$$P_I \subseteq \{x \in \mathbb{R}^n : c^T x \leq \lfloor \delta \rfloor\}.$$

Diese Beobachtung legt nahe, alle Stützhyperebenen mit ganzzahliger linker Seite zu nehmen und die rechte Seite zu runden, um eine schärfere Formulierung zu erhalten. Definiere dazu

(3.3)
$$P^1 := \{x \in \mathbb{R}^n : c^T x \leq \lfloor \delta \rfloor \quad \text{für alle Stützhyperebenen } \{x \in \mathbb{R}^n : c^T x = \delta\} \text{ von } P \text{ mit } c \text{ ganzzahlig}\}.$$

Auf den ersten Blick ist nicht klar, ob P^1 wieder ein Polyeder ist, weil es unendlich viele Stützhyperebenen gibt. Wir werden jedoch beweisen, dass P^1 tatsächlich wieder ein Polyeder ist. Damit können wir das Verfahren fortsetzen und dasselbe Spiel mit P^1 machen. Definiere

$$P^0 := P \text{ und } P^{t+1} := (P^t)^1. \tag{3.4}$$

Die folgenden Beziehungen sind offensichtlich.

$$P = P^0 \supseteq P^1 \supseteq \ldots \supseteq P_I. \tag{3.5}$$

Damit stellt sich die Frage, ob dieses Verfahren endlich ist. In der Tat, es ist es, und wir werden zeigen, dass $P^t = P_I$ für ein $t \in \mathbb{N}$. Das heißt, dass P^t unser Ziel liefert, nämlich eine Beschreibung P_I in Form von linearen Ungleichungen.

Wir beginnen mit dem Beweis, dass P^1 ein Polyeder ist. Wir erinnern uns aus Satz 2.37, dass jedes rationale Polyeder durch ein TDI System der Form $Ax \leq b$ mit A ganzzahlig beschrieben werden kann. Und in der Tat, runden wir die rechte Seite dieses Systems, so erhalten wir P^1.

Satz 3.6 *Sei $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ mit $Ax \leq b$ TDI und A ganzzahlig. Dann gilt $P^1 = \{x \in \mathbb{R}^n : Ax \leq \lfloor b \rfloor\}$, d.h. P^1 ist ein Polyeder.*

Beweis. Im Falle $P = \emptyset$ ist nichts zu beweisen. Sei $P \neq \emptyset$. Offensichtlich gilt $P^1 \subseteq \{x \in \mathbb{R}^n : Ax \leq \lfloor b \rfloor\}$ (wähle als Stützhyperebene $\{x \in \mathbb{R}^n : A_{i,\cdot}x \leq b_i\}$). Um die Umkehrung zu beweisen, sei $\{x \in \mathbb{R}^n : c^T x = \delta\}$ eine Stützhyperebene von P mit $P \subseteq \{x \in \mathbb{R}^n : c^T x \leq \delta\}$ und c ganzzahlig, $\delta \in \mathbb{Q}$. Aus dem Dualitätssatz der Linearen Optimierung wissen wir

$$\delta = \max\{c^T x : Ax \leq b,\, x \in \mathbb{R}^n\} = \min\{y^T b : A^T y = c,\, y \geq 0\}.$$

Da $Ax \leq b$ TDI ist, gibt es eine ganzzahlige optimale Lösung y^* von $\min\{y^T b : A^T y = c,\, y \geq 0\}$. Da x die Ungleichung $Ax \leq \lfloor b \rfloor$ erfüllt, gilt $c^T x = (A^T y^*)^T x = (y^*)^T (Ax) \leq (y^*)^T \lfloor b \rfloor = \lfloor (y^*)^T \lfloor b \rfloor \rfloor \leq \lfloor (y^*)^T b \rfloor = \lfloor \delta \rfloor$. Es folgt

$$\{x \in \mathbb{R}^n : Ax \leq \lfloor b \rfloor\} \subseteq \{x \in \mathbb{R}^n : c^T x \leq \lfloor \delta \rfloor\}.$$

Also gilt

$$\{x \in \mathbb{R}^n : Ax \leq \lfloor b \rfloor\} \subseteq \bigcap_{c,\delta} \{x \in \mathbb{R}^n : c^T x \leq \lfloor \delta \rfloor\} = P^1,$$

wobei die Schnittmenge über allen Stützhyperebenen $\{x \in \mathbb{R}^n : c^T x = \delta\}$ mit ganzzahligem c und $P \subseteq \{x \in \mathbb{R}^n : c^T x \leq \delta\}$ genommen wird. □

Folgerung 3.7 *Sei F eine Seitenfläche von P. Dann gilt $F^1 = P^1 \cap F$.*

Beweis. Sei $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ mit $Ax \leq b$ TDI und A ganzzahlig. Sei $F = \{x \in P : c^T x = \delta\}$ eine Seitenfläche von P mit c, δ ganzzahlig und $P \subseteq \{x \in \mathbb{R}^n : c^T x \leq \delta\}$. Satz 2.34 impliziert, dass das System $\{Ax \leq b,\, c^T x \leq \delta\}$ ebenso TDI ist. Also ist nach Korollar 2.35 $\{Ax \leq b,\, c^T x = \delta\}$ auch TDI. Wir erhalten nach Satz 3.6

$$\begin{aligned} P^1 \cap F &= P^1 \cap P \cap \{x \in \mathbb{R}^n : c^T x = \delta\} \\ &= \{x \in \mathbb{R}^n : Ax \leq \lfloor b \rfloor,\, c^T x = \delta\} \\ &= \{x \in \mathbb{R}^n : Ax \leq \lfloor b \rfloor,\, c^T x \leq \lfloor \delta \rfloor,\, -c^T x \leq \lfloor -\delta \rfloor\} \\ &= F^1. \end{aligned}$$

□

Bemerkung 3.8

(a) F^1 ist eine (möglicherweise leere) Seitenfläche von P^1, da $F^1 = P^1 \cap F = P^1 \cap P \cap \{x \in \mathbb{R}^n : c^T x = \delta\} = P^1 \cap \{x \in \mathbb{R}^n : c^T x = \delta\}$ (beachte, dass $P^1 \subseteq P \subseteq \{x \in \mathbb{R}^n : c^T x \leq \delta\}$).

(b) Die wiederholte Anwendung von Korollar 3.7 liefert

$$F^t = P^t \cap F, \text{ für } t = 1, 2, \ldots$$

Nun haben wir alles zusammen, um die Endlichkeit der Rundeverfahrens zu zeigen.

Satz 3.9 *Sei P ein rationales Polyeder. Dann gibt es eine natürliche Zahl $t \in \mathbb{N}$, so dass $P^t = P_I$.*

Beweis. Ist $P = \emptyset$, dann folgt $P^0 = P = P_I$. Wir können also $P \neq \emptyset$ annehmen. Wir beweisen die Behauptung mittels Induktion nach der Dimension d von P.

Ist $d = 0$, dann gilt $P = \{\bar{x}\}$ für ein $\bar{x} \in \mathbb{R}^n$. Ist $\bar{x} \in \mathbb{Z}^n$, so erhalten wir $P^0 = P = P_I$. Im anderen Fall ist ($\bar{x} \notin \mathbb{Z}^n$) $P^1 = \emptyset = P_I$. Es sei $d > 0$ und die Behauptung sei wahr für alle Polyeder von kleinerer Dimension. Wir zeigen zunächst, dass wir o.B.d.A. annehmen können, dass P volldimensional ist.

Sei $\{x \in \mathbb{R}^n : Ax = b\}$ die affine Hülle von P, d.h. $P \subseteq \{x \in \mathbb{R}^n : Ax = b\}$. O.B.d.A. ist A eine ganzzahlige Matrix mit vollem Zeilenrang, d.h. der Zeilenrang von A ist $n - d$.

Falls $\{x \in \mathbb{R}^n : Ax = b\}$ keine ganzzahlige Lösung hat, dann gibt es ein $y \in \mathbb{Q}^m$ mit $c := A^T y \in \mathbb{Z}^n$ und $\delta := b^T y \notin \mathbb{Z}$ nach Satz 2.27. Jedes $x \in P$ erfüllt $Ax = b$ und damit gilt $c^T x = (A^T y)^T x = y^T Ax = y^T b = \delta$. Also ist $\{x \in \mathbb{R}^n : c^T x = \delta\}$ eine Stützhyperebene von P. Wir folgern

$$P^1 \subseteq \{x \in \mathbb{R}^n : c^T x \leq \lfloor \delta \rfloor, c^T x \geq \lceil \delta \rceil\} = \emptyset$$

woraus wiederum $P_I \subseteq P^1 = \emptyset$ folgt.

Nun sei $\hat{x}$ eine ganzzahlige Lösung von $\{x \in \mathbb{R}^n : Ax = b\}$. Satz 3.9 ist invariant unter Verschiebung um den Vektor $\hat{x}$ und somit können wir $\text{aff}(P) = \{x \in \mathbb{R}^n : Ax = 0\}$ annehmen. Aus Satz 2.25 und Bemerkung 2.26 (a) sowie der Tatsache, dass der Zeilenrang von A gleich $n-d$ ist, folgt, dass es eine reguläre unimodulare Matrix U gibt mit $AU = [B, 0]$ und $B \in \mathbb{Z}^{(n-d)\times(n-d)}$ ist regulär. Da U regulär und unimodular ist, ist auch U^{-1} unimodular und die Variablentransformation

$$x = Uz.$$

verletzt nicht die Gültigkeit des Satzes, insbesondere gilt $x \in \mathbb{Z}^n \Leftrightarrow z \in \mathbb{Z}^n$. Wir können annehmen, dass $\text{aff}(P) = \{z \in \mathbb{R}^n : [B, 0]z = 0\} = \{0\}^{n-d} \times \mathbb{R}^d$ gilt. Jede Stützhyperebene $H = \{x \in \mathbb{R}^n : c^T x = \delta\}$ von P wird in die Standardform $H' = \{x \in \mathbb{R}^n : \sum_{i=1}^{n-d} 0x_i + \sum_{i=n-d+1}^{n} c^T x_i = \delta\}$ überführt, indem geeignete Vielfache der Zeilen von $[B, 0]$ zu c addiert werden. Bei der Konstruktion von P^1 können wir uns auf die Stützhyperebenen der Form H' beschränken. Wir können also $n - d = 0$ annehmen, d.h. P ist volldimensional.

Wir behaupten, es gibt eine ganzzahlige Matrix W, einen rationalen Vektor w und einen ganzzahligen Vektor w', so dass $P = \{x \in \mathbb{R}^n : Wx \leq w\}$ und $P_I = \{x \in \mathbb{R}^n : Wx \leq w'\}$ ist. Um dies nachzuvollziehen, sei $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ und $P_I = \{x \in \mathbb{R}^n : Cx \leq b'\}$ mit ganzzahligen Matrizen A und C. Definiere $W := \begin{bmatrix} A \\ C \end{bmatrix}$. Für jeden Zeilenindex von W liefert die Wahl $w_i := \max\{W_{i,\cdot}x : x \in P\}$ und $w'_i := \max\{W_{i,\cdot}x : x \in P_I\}$ das Gewünschte. Beachte, dass wir annehmen können, dass P bereits zu Beginn in dieser Form gegeben ist, da in der Konstruktion von P^1 *alle* Stützhyperebenen von P betrachtet werden, insbesondere alle Zeilen der Matrix W.

Betrachte eine Ungleichung $a^Tx \leq \beta'$ des Systems $Wx \leq w'$. Wir behaupten, dass es eine natürliche Zahl $s \in \mathbb{N}$ gibt mit $P^s \subseteq \{x \in \mathbb{R}^n : a^Tx \leq \beta'\}$.
Nehmen wir das Gegenteil an. Sei $a^Tx \leq \beta$ die zugehörige Ungleichung von $Wx \leq w$. Wegen $P^1 \subseteq \{x \in \mathbb{R}^n : a^Tx \leq \lfloor\beta\rfloor\}$ gibt es ein $\beta'' \in \mathbb{Z}$ und $r \in \mathbb{N}$, so dass gilt:

$$\begin{aligned} &\beta' < \beta'' \leq \lfloor\beta\rfloor \text{ und} \\ &P^u \subseteq \{x \in \mathbb{R}^n : a^Tx \leq \beta''\} \\ &P^u \not\subseteq \{x \in \mathbb{R}^n : a^Tx \leq \beta'' - 1\} \text{ für alle } u \geq r. \end{aligned}$$

Auf Grund der Wahl von r ist $\{x \in \mathbb{R}^n : a^Tx = \beta''\}$ eine Stützhyperebene von P^r (beachte, dass $P \not\subseteq \{x \in \mathbb{R}^n : a^Tx \leq \beta''\}$, da P volldimensional) und damit hat $F := P^r \cap \{x \in \mathbb{R}^n : a^Tx = \beta''\}$ eine geringere Dimension als n. Weiter folgern wir aus $P_I \subseteq \{a^Tx \leq \beta'\}$, dass $F \cap \mathbb{Z}^n = \emptyset$ gilt. Also existiert nach der Induktionsannahme ein $q \in \mathbb{N}$ mit $F^q = \emptyset$. Aus Bemerkung 3.8 (b) folgt

$$\begin{aligned} \emptyset = F^q &= \left(P^r \cap \{x \in \mathbb{R}^n : a^Tx = \beta''\}\right)^q \\ &= P^{r+q} \cap \{x \in \mathbb{R}^n : a^Tx = \beta''\}. \end{aligned}$$

Es gilt also $P^{r+q} \subseteq \{x \in \mathbb{R}^n : a^Tx < \beta''\}$, was heißt $P^{r+q+1} \subseteq \{x \in \mathbb{R}^n : a^Tx \leq \beta'' - 1\}$ und dies ist ein Widerspruch zur Wahl von β'' und r.
Da $a^Tx \leq \beta'$ beliebig gewählt war und das System $Wx \leq w'$ endlich ist, haben wir Satz 3.9 bewiesen. □

Betrachten wir uns noch einmal, was wir bisher gezeigt haben. Die Vorgehensweise, um eine lineare Beschreibung des ganzzahligen Polyeders $P_I = \text{conv}\{x \in \mathbb{Z}^n : Ax \leq b\}$ zu erreichen, geht wie folgt. Wir beginnen mit der linearen Relaxierung $P^0 = P = \{x \in \mathbb{R}^n : Ax \leq b\}$ von P_I. Als nächstes betrachten wir jede Stützhyperebene von P, deren linke Seite ganzzahlig ist und runden die rechte Seite nach unten zur nächsten ganzen Zahl ab. Diese Prozedur, für alle solche Stützhyperebenen durchgeführt, ergibt das Polyeder P^1. Satz 3.6 zeigt, dass keine Notwendigkeit besteht, alle Stützhyperebenen zu testen. Alles, was wir benötigen, ist ein TDI-System $Dx \leq d$, das P beschreibt. Für dieses TDI-System müssen wir den Vektor der rechten Seite d nach unten abrunden. Im Beweis von Satz 2.37 haben wir das TDI-System für ein rationales Polyeder P explizit konstruiert, indem wir für jede Seitenfläche F von P eine Hilbert-Basis des Kegels $\text{cone}\,(A_{\text{eq}(F)})$

erzeugen. Letztendlich erhalten wir mit einem Verfahren zur Konstruktion einer Hilbert-Basis für einen polyedrischen Kegel insgesamt einen Algorithmus zur Berechnung von P^1. Nach Satz 3.9 müssen wir diesen Algorithmus nur eine endliche Zahl von Schritten durchführen, um eine lineare Beschreibung von P_I zu erzielen.

Dennoch ist die gesamte Prozedur kaum praktikabel. Zunächst ist die Zahl $t \in \mathbb{N}$ in Satz 3.9 möglicherweise exponentiell in der Kodierungslänge der Eingabegrößen A, b (siehe Übung). Zweitens müssen wir in jeder Iteration i Hilbert-Basen für alle Kegel bestimmen, die von den Seitenflächen von P^{i-1} erzeugt werden. Im Allgemeinen ist nicht nur die Zahl der Seitenflächen exponentiell, wir können auch die Hilbert-Basen nicht in polynomialer Zeit berechnen.

Andererseits ist zum Lösen von (3.1) eine vollständige Beschreibung von $P_I = \text{conv}\{x \in \mathbb{Z}^n : Ax \leq b\}$ nicht notwendig. Alles, was wir benötigen, ist das Finden einer optimalen Lösung. Anders ausgedrückt, wir sind nur an einer Seitenfläche interessiert, nämlich an der, die die optimalen Lösungen enthält. Ja sogar für diese Seitenfläche ist es nicht unbedingt notwendig, eine Hilbert-Basis für den zugehörigen Kegel zu bestimmen. Wir stellen uns also die Frage, ob es möglich ist, ganzzahlige Vektoren in diesem Kegel nach Bedarf zu bestimmen. Wir nehmen an, wir beginnen mit der Lösung der LP-Relaxierung $\max\{c^T x : Ax \leq b\}$. Sei x^* eine optimale Lösung, die nicht ganzzahlig ist. Die zentrale Frage besteht darin, einen ganzzahligen Vektor in $C(A_{\text{eq}(x^*)})$ zu finden, der die momentane optimale Lösung abschneidet, d.h. finde $d \in \mathbb{Z}^n \cap C(A_{\text{eq}(x^*)})$ mit $d^T x^* \notin \mathbb{Z}$.

Gomorys algorithmischer Ansatz

[19] schlug einen systematischen Weg vor, um einen solchen Vektor d durch Auswertung der Information aus dem Simplexalgorithmus zu erhalten. Wir nehmen an, A und b seien ganzzahlig. Wir nehmen weiterhin an, dass (3.1) in Standardform

$$\begin{aligned} \max \quad & c^T x \\ & Ax = b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

mit $P = \{x \in \mathbb{R}^n : Ax = b,\, x \geq 0\}$ und $P_I = \text{conv}\{x \in \mathbb{Z}^n : Ax = b,\, x \geq 0\}$ ist. Dies kann durch Aufspaltung von x in x^+ und x^- mit $x = x^+ - x^-$, $x^+, x^- \geq 0$ erreicht werden, wenn die Nichtnegativitäts-Bedingungen nicht in das System $Ax \leq b$ aufgenommen werden, wobei Schlupfvariablen für die kleiner oder gleich Bedingungen eingeführt werden. Beachte, dass für die Schlupfvariablen ebenfalls Ganzzahligkeit gefordert werden kann, da alle Werte ganzzahlig sind.
Lösen wir die LP-Relaxierung

$$\begin{aligned} \max \quad & c^T x \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

mittels der Simplexmethode, so erhalten wir eine optimale Lösung x^* und eine optimale Basis B mit $B \subseteq \{1, \ldots, n\}$, $|B| = m$ und A_B regulär. Über die Basis B drücken sich die Werte von x^* aus als

$$\begin{aligned} x^*_B &= A_B^{-1} b - A_B^{-1} A_N x^*_N \\ x^*_N &= 0. \end{aligned} \tag{3.6}$$

Ist x^* ganzzahlig, haben wir eine optimale Lösung für (3.1). Im anderen Fall ist einer der Werte x^*_B gebrochen. Sei $i \in B$ ein Index mit $x^*_i \notin \mathbb{Z}$. Da jede zulässige ganzzahlige Lösung (3.6) erfüllt, gilt

$$A^{-1}_{i,\cdot} b - \sum_{j \in N} A^{-1}_{i,\cdot} A_{\cdot,j} x_j \in \mathbb{Z} \tag{3.7}$$

für alle ganzzahligen Lösungen x. (3.7) gilt auch, wenn wir ganzzahlige Vielfache von x_j, $j \in N$, oder eine ganze Zahl zu $\bar{A}^{-1}_{i,\cdot} b$ addieren. Wir erhalten

$$f(A^{-1}_{i,\cdot} b) - \sum_{j \in N} f(A^{-1}_{i,\cdot} A_{\cdot,j}) x_j \in \mathbb{Z} \tag{3.8}$$

für alle ganzzahligen Lösungen x, wobei $f(\alpha) = \alpha - \lfloor \alpha \rfloor$ für $\alpha \in \mathbb{R}$ ist. Da $0 \leq f(\cdot) < 1$ und $x \geq 0$ ist, folgt

$$f(A^{-1}_{i,\cdot} b) - \sum_{j \in N} f(A^{-1}_{i,\cdot} A_{\cdot,j}) x_j \leq 0.$$

Also ist die Ungleichung

$$\sum_{j \in N} f(A^{-1}_{i,\cdot} A_{\cdot,j}) x_j \geq f(A^{-1}_{i,\cdot} b) \tag{3.9}$$

gültig für alle zulässigen ganzzahligen Lösungen x. Weiterhin ist sie von der momentanen LP-Lösung x^* verletzt, da $x^*_N = 0$ und $f(A^{-1}_{i,\cdot} b) = f(x^*_i) > 0$ ist.
Es stellt sich heraus, dass (3.9) in der Tat eine Stützhyperebene von P mit ganzzahliger linker Seite ist. Um dies zu sehen, subtrahiere $A^{-1}_{i\cdot} Ax = A^{-1}_{i\cdot} b \Leftrightarrow x_i + \sum_{j \in N} A^{-1}_{i\cdot} A_{\cdot j} x_j = A^{-1}_{i\cdot} b$ von (3.9) und man erhält

$$x_i + \sum_{j \in N} \lfloor A^{-1}_{i\cdot} A_{\cdot j} \rfloor x_j \leq \lfloor A^{-1}_{i\cdot} b \rfloor,$$

und damit, wenn die rechte Seite nicht gerundet ist, eine Stützhyperebene von P mit ganzzahliger linker Seite.

Fügt man überdies diese Ungleichung zu dem System $Ax = b, x \geq 0$ hinzu, so erhält sich die Eigenschaft, dass alle Werte ganzzahlig sind. Also kann die Schlupfvariable, die für die neue Ungleichung eingeführt werden muss, ebenso als ganzzahlig angesehen werden und so das Verfahren iteriert werden. [20] zeigt, dass mit einer bestimmten Wahl der erzeugenden Zeile für die Schnitte ein endlicher Algorithmus vorliegt, d.h. nachdem eine endliche Zahl an Ungleichungen hinzugefügt wurde, wird eine ganzzahlige Lösung gefunden. Dies stellt einen alternativen Beweis für Satz 3.9 dar.

Beispiel 3.14 *Betrachte das folgende IP*

$$\begin{array}{lrl} \max & x_2 & \\ & 4x_1 \; + x_2 & \leq 4 \\ & -4x_1 \; + x_2 & \leq 0 \\ & x_1, x_2 & \geq 0 \\ & x_1, x_2 & \in \mathbb{Z}. \end{array}$$

Hinzufügen von Schlupfvariablen ergibt

$$\begin{array}{lrlrl} \max & x_2 & & & \\ & 4x_1 \; + x_2 & + x_3 & & = 4 \\ & -4x_1 \; + x_2 & & + x_4 & = 0 \\ & & x_1, x_2, x_3, x_4 & & \geq 0 \\ & & x_1, x_2, x_3, x_4 & & \in \mathbb{Z}. \end{array}$$

Die optimale Basis für die LP-Relaxierung ist $B = \{1, 2\}$ *mit*

$$A_B^{-1} = \begin{bmatrix} \frac{1}{8} & -\frac{1}{8} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}.$$

Also ist $x_B^* = \binom{1/2}{2}$. *Die einzige gebrochene Komponente ist* x_1 *und für* $i = 1$ *wird (3.7) zu*

$$\tfrac{1}{2} - \tfrac{1}{8}x_3 + \tfrac{1}{8}x_4 \in \mathbb{Z},$$

was für alle ganzzahligen Lösungen x *gilt. Wir fügen* x_4 *zur obigen Gleichheit hinzu, um (3.9) zu erhalten:*

$$\tfrac{1}{8}x_3 + \tfrac{7}{8}x_4 \geq \tfrac{1}{2}.$$

Um zu zeigen, dass die Ungleichung zu einer Stützhyperebene gehört, subtrahieren wir $x_1 + 0x_2 + \frac{1}{8}x_3 - \frac{1}{8}x_4 = \frac{1}{2}$ *und erhalten*

$$-x_1 + x_4 \geq 0 \; = \; \lfloor \tfrac{1}{2} \rfloor$$

Drücken wir diese Ungleichung in den Originalvariablen x_1, x_2 aus, erhalten wir durch Ersetzen der Schlupfvariable x_4

$$-3x_1 + x_2 \leq 0.$$

Dies ist eine gültige Ungleichung für $P_I = conv\{x \in \mathbb{Z}^2_+ : 4x_1 + x_2 \leq 4, -4x_1 + x_2 \leq 0\}$, siehe (3.9).

Gomorys gemischt-ganzzahlige Schnitte

Wir wollen uns schließlich noch mit dem Fall beschäftigen, dass wir kein rein ganzzahliges Programm haben, sondern ein gemischt-ganzzahliges Programm. In diesem Fall funktionieren die Ideen von Gomory und Chvátal nicht. Chvátal's Ansatz funktioniert nicht, da die rechte Seite in (3.3) nicht abgerundet werden kann. Gomory's Ansatz versagt, da es nicht mehr länger möglich ist, ganzzahlige Vielfache auf kontinuierliche Variablen zu addieren, um (3.8) aus (3.7) zu erhalten. So hat beispielsweise $\frac{1}{3} + \frac{1}{3}x_1 - 2x_2 \in \mathbb{Z}$ mit $x_1 \in \mathbb{Z}_+, x_2 \in \mathbb{R}_+$ eine größere Lösungsmenge (z.B. $\binom{1}{1/3}$) als $\frac{1}{3} + \frac{1}{3}x_1 \in \mathbb{Z}$. Wir können also nicht mehr garantieren, dass die Koeffizienten der kontinuierlichen Variablen nichtnegativ sind und damit die Gültigkeit von (3.9) nachweisen. Dennoch kann man mittels des folgenden *disjunktiven Arguments* gültige Ungleichungen erzeugen.

Beobachtung 3.15 *Sei $(a^k)^T x \leq \alpha^k$ eine gültige Ungleichung für ein Polyeder $P^k \subseteq \mathbb{R}^n_+$ für $k = 1, 2$. Dann ist*

$$\sum_{i=1}^{n} \min(a^1_i, a^2_i) x_i \leq \max(\alpha^1, \alpha^2)$$

gültig für $P^1 \cup P^2$ und $conv(P^1 \cup P^2)$.

Diese Beobachtung kann auf verschiedene Weise gültige Ungleichungen für den gemischt-ganzzahligen Fall erzeugen. Wir erläutern einen dieser Wege, nämlich Gomory's gemischt ganzzahlige Schnitte.
Wir betrachten wieder die Situation in (3.7), wobei $x_i, i \in B$, eine ganze Zahl sein soll. Wir benutzen die folgenden Abkürzungen $\bar{a}_j = A^{-1}_{i.} A_{.j}$, $\bar{b} = A^{-1}_{i.} b$, $f_j = f(\bar{a}_j)$, $f_0 = f(\bar{b})$, und $N^+ = \{j \in N : \bar{a}_j \geq 0\}$ sowie $N^- = N \setminus N^+$. Der Ausdruck (3.7) ist äquivalent zu $\sum_{j \in N} \bar{a}_j x_j = f_0 + k$ für ein $k \in \mathbb{Z}$. Wir unterscheiden nun die zwei Fälle $\sum_{j \in N} \bar{a}_j x_j \geq f_0 \geq 0$ und $\sum_{j \in N} \bar{a}_j x_j \leq f_0 - 1 < 0$. Im ersten Fall muss

$$\sum_{j \in N^+} \bar{a}_j x_j \geq f_0$$

gelten. Im zweiten Fall erhalten wir $\sum_{j \in N^-} \bar{a}_j x_j \leq f_0 - 1$, was zu

$$-\frac{f_0}{1 - f_0} \sum_{j \in N^-} \bar{a}_j x_j \geq f_0$$

äquivalent ist. Wir wenden nun Beobachtung 3.15 auf die Zerlegung $P^1 = P \cap \{x : \sum_{j \in N} \bar{a}_j x_j \geq 0\}$ und $P^2 = P \cap \{x : \sum_{j \in N} \bar{a}_j x_j \leq 0\}$ an und erhalten die gültige Ungleichung

$$\sum_{j \in N^+} \bar{a}_j x_j - \frac{f_0}{1 - f_0} \sum_{j \in N^-} \bar{a}_j x_j \geq f_0. \tag{3.10}$$

Diese Ungleichung kann auf folgende Weise verschärft werden. Man beachte, dass die Herleitung von (3.10) immer noch gültig ist, wenn wir ganzzahlige Vielfache zu ganzzahligen Variablen addieren. Auf diese Weise können wir jede ganzzahlige Variable entweder in die Menge N^+ oder in die Menge N^- einordnen. Ist eine Variable in N^+, so ist der Koeffizient in (3.10) gleich $\bar{a}_j$ und somit ist der beste mögliche Koeffizient nach der Addition von ganzzahligen Vielfachen $f_j = f(\bar{a}_j)$. In N^- ist der Koeffizient in (3.10) gleich $-\frac{f_0}{1-f_0}\bar{a}_j$ und somit ist $\frac{f_0(1-f_j)}{1-f_0}$ die beste Wahl. Insgesamt erhalten wir den bestmöglichen Koeffizienten durch die Wahl von $\min(f_j, \frac{f_0(1-f_j)}{1-f_0})$. Dies führt auf Gomory's gemischt-ganzzahligen Schnitt [18]

$$\begin{array}{llllll} \sum\limits_{\substack{j:\, f_j \leq f_0 \\ j \text{ ganzzahlig}}} & f_j x_j & + & \sum\limits_{\substack{j:\, f_j > f_0 \\ j \text{ ganzzahlig}}} & \frac{f_0(1-f_j)}{1-f_0} x_j & + \\ \sum\limits_{\substack{j \in N^+ \\ j \text{ nicht ganzzahlig}}} & \bar{a}_j x_j & - & \sum\limits_{\substack{j \in N^- \\ j \text{ nicht ganzzahlig}}} & \frac{f_0}{1-f_0} \bar{a}_j x_j & \geq \quad f_0. \end{array} \tag{3.11}$$

[18] zeigt, dass ein Algorithmus, der auf iterativer Hinzufügung dieser Ungleichungen beruht, $\min\{c^T x : x \in X\}$ mit $X = \{x \in \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p} : Ax = b\}$ in einer endlichen Anzahl von Schritten löst, falls $c^T x \in \mathbb{Z}$ für alle $x \in X$ gilt.
Man kennt in der Literatur eine ganze Reihe weitere Ungleichungen, die unabhängig von einer bestimmten Problemstruktur sind. Unter diesen findet man beispielsweise Mixed Integer Rounding Schnitte (MIR) [29] und sogenannte Lift-and-Project Schnitte [2].

3.1.2 Ungleichungen mit Struktur

Wir haben uns soeben mit gültigen Ungleichungen für allgemeine IP's und MIP's beschäftigt. Richten wir unser Augenmerk auf eine einzelne Nebenbedingung oder eine kleine Teilmenge von solchen Nebenbedingungen, so kann ein allgemeines Problem eine gewisse "lokale" Struktur enthalten. So können beispielsweise alle Variablen in einer Nebenbedingung 0/1-Variablen sein, oder ein kleiner Teil eines MIP kann ein Flussproblem in einem Netzwerk sein. Wir suchen hier nach Wegen, um strengere Ungleichungen unter Ausnutzung solcher lokaler Strukturen zu erhalten.

Rucksack und Cover Ungleichungen

Das Konzept eines Covers wurde in der Literatur häufig benutzt, um gültige Ungleichungen für (gemischt) ganzzahlige Mengen zu erzeugen. In diesem Abschnitt zeigen wir zunächst, wie wir dieses Konzept nutzen können, um Cover Ungleichungen für die 0/1-Rucksackmenge zu erzeugen. Wir überlegen dann, wie wir diese Ungleichungen auf komplexere gemischt ganzzahlige Mengen ausdehnen können.

Definition 3.18 *Wir betrachten das* 0/1*-Rucksackpolytop*

$$P_K(N, a, b) = \{x \in \{0,1\}^N : \sum_{j \in N} a_j x_j \leq b\}$$

mit nichtnegativen Koeffizienten, d.h. es ist $a_j \geq 0$ *für* $j \in N$ *und* $b \geq 0$*. Eine Menge* $C \subseteq N$ *heißt* Cover*, wenn gilt:*

$$\lambda = \sum_{j \in C} a_j - b > 0. \tag{3.12}$$

Zusätzlich nennt man den Cover C minimal*, wenn* $a_j \geq \lambda$ *für alle* $j \in C$*.*

Mit jedem Cover C können wir eine einfache gültige Ungleichung identifizieren, die aussagt, dass nicht alle Variablen x_j für $j \in C$ gleichzeitig zu eins gesetzt werden können.

Satz 3.20 *[1, 22, 32, 37]*
Sei $C \subseteq N$ *ein Cover. Die* Coverungleichung

$$\sum_{j \in C} x_j \leq |C| - 1 \tag{3.13}$$

ist gültig für $P_K(N, a, b)$*. Ist überdies* C *minimal, dann definiert die Ungleichung* (3.13) *eine Facette von* $P_K(C, a, b)$*.*

Beweis. Die Gültigkeit ist offensichtlich. Um zu zeigen, dass diese eine Facette induziert, nehmen wir an, es gibt eine facetten-definierende Ungleichung $b^T x \leq \beta$ mit $\{x \in P : \sum_{j \in C} x_j = |C| - 1\} \subseteq F_b := \{x \in P : b^T x = \beta\}$. Mit $C_i = C \setminus \{i\}$ folgt $\chi^{C_i} \in F_b$ für alle $i \in C$. Damit gilt $0 = b^T \chi^i - b^T \chi^j = b_i - b_j$ und damit $b_i = b_j$ für alle $i, j \in C$. Also ist $b^T x \leq \beta$ bis auf ein positives Vielfaches die Coverungleichung. □
Ist ein Cover C nicht minimal, dann kann man leicht erkennen, dass die zugehörige Coverungleichung redundant ist, d.h. sie ist die Summe einer minimalen Coverungleichung und einigen Bedingungen, die sich aus oberen Schranken ergeben.

Beispiel 3.22 *Betrachte die* 0/1*-Rucksackmenge*

$$P_K = \{x \in \{0,1\}^6 : 5x_1 + 5x_2 + 5x_3 + 5x_4 + 3x_5 + 8x_6 \leq 17\}.$$

$C = \{1,2,3,4\}$ *ist ein minimaler Cover für* P_K *und die zugehörige Coverungleichung*

$$x_1 + x_2 + x_3 + x_4 \leq 3$$

definiert eine Facette von $conv(\{x \in \{0,1\}^4 : 5x_1 + 5x_2 + 5x_3 + 5x_4 \leq 17\})$.

Es gibt viele weitere Klassen von Ungleichungen für das Rucksack-Polytop, u.a. sog. $(1,k)$-Konfigurations- oder "Extended Weight"-Ungleichungen. Darüber hinaus kann die sog. *Liftingmethode* verwendet werden, um nicht facettendefinierende Cover Ungleichungen zu verschärfen. Man erhält die Klasse der "Lifted Cover" Ungleichungen, auf die wir hier nicht näher eingehen wollen. Weiterführende Literatur hierzu ist zu finden in [33, 36, 28].
Das Separieren der Coverungleichungen ist $\mathcal{NP}$-schwer ([11, 27]), so dass hier im allgemeinen auf Heuristiken zurückgegriffen wird.

Das Konzept eines Covers ist auch sinnvoll bei den Studien der polyedrischen Struktur von Problemen, die sowohl 0/1 als auch stetige Variablen enthalten. Betrachte die gemischte 0/1-Rucksackmenge

$$S = \{\binom{x}{s} \in \{0,1\}^N \times \mathbb{R}_+ : \sum_{j \in N} a_j x_j \leq b + s\}$$

mit nichtnegativen Koeffizienten, d.h. $a_j \geq 0$ für $j \in N$ und $b \geq 0$.

Satz 3.23 *[29] Sei* $C \subseteq N$ *ein Cover, d.h.* C *ist eine Teilmenge von* N*, die* (3.12) *erfüllt. Die Ungleichung*

$$\sum_{j \in C} \min(a_j, \lambda)\, x_j \leq \sum_{j \in C} \min(a_j, \lambda) - \lambda + s \tag{3.14}$$

ist gültig für S*. Überdies definiert die Ungleichung* (3.14) *eine Facette von* $conv(S_C)$*, wobei* $S_C = S \cap \{x : x_j = 0,\ j \in N \backslash C\}$ *gilt.*

Beweis. Wir zeigen die Gültigkeit der Ungleichung. Zum Beweis der Facetten-Eigenschaft verweisen wir auf [29].
Sei $\binom{x}{s} \in S$ beliebig. Sei $C_\lambda = \{j \in C : \lambda \leq a_j\}$ und $C_a = C \setminus C_\lambda$. Wir unterscheiden folgende Fälle:

1. $C_\lambda = \emptyset$. Dann gilt

$$\sum_{j \in C} a_j x_j \leq b + s = \sum_{j \in C} a_j - \lambda + s$$

2. $C_\lambda \neq \emptyset$.

   (a) $x_j = 0$ für ein $j \in C_\lambda$. Dann gilt

$$\begin{aligned}\sum_{j\in C_\lambda} \lambda x_j + \sum_{j\in C_a} a_j x_j &\leq (|C_\lambda| - 1)\lambda + \sum_{j\in C_a} a_j x_j \\ &\leq (|C_\lambda| - 1)\lambda + \sum_{j\in C_a} a_j + s.\end{aligned}$$

   (b) $x_j = 1$ für alle $j \in C_\lambda$.

$$\begin{aligned}\sum_{j\in C} \min(a_j, \lambda)\, x_j &= |C_\lambda|\lambda + \sum_{j\in C_a} a_j x_j \\ &\leq |C_\lambda|\lambda + s - \sum_{j\in C_\lambda} a_j + b \\ &= |C_\lambda|\lambda + s - \sum_{j\in C_\lambda} a_j + \sum_{j\in C} a_j - \lambda \\ &= |C_\lambda|\lambda + \sum_{j\in C_a} a_j - \lambda + s \\ &= \sum_{j\in C} \min(a_j, \lambda) - \lambda + s.\end{aligned}$$

□

Beachte, dass jeder Cover C eine Coverungleichung impliziert, die eine Facette von $\mathrm{conv}(S_C)$ definiert. Dies ist anders als im rein ganzzahligen Fall, wo nur minimale Cover Facetten induzieren.

Beispiel 3.25 *Betrachte die gemischte* 0/1*-Rucksackmenge*

$$S = \{(x, s) \in \{0, 1\}^6 \times \mathbb{R}_+ : 5x_1 + 5x_2 + 5x_3 + 5x_4 + 3x_5 + 8x_6 \leq 17 + s\}.$$

Wählen wir $C' = \{1, 2, 3, 6\}$ *als (nichtminimalen) Cover für* S*, so definiert die zugehörige Coverungleichung*

$$5x_1 + 5x_2 + 5x_3 + 6x_6 \leq 15 + s$$

eine Facette von $\mathrm{conv}(\{(x, s) \in \{0, 1\}^4 \times \mathbb{R}_+ : 5x_1 + 5x_2 + 5x_3 + 8x_6 \leq 17 + s\})$.

Das Set-Packing Polytop

Ganzzahlige und gemischt ganzzahlige Probleme enthalten oft einige Nebenbedingungen, die nur 0/1-Koeffizienten enthalten. Viele Preprocessingroutinen für ganzzahlige Probleme erzeugen automatisch logische Ungleichungen der Form

$x_i + x_j \leq 1, x_i \leq x_j$, Coverungleichungen, usw. Dies führt auf das Studium von ganzzahligen Programmen mit 0/1-Matrizen.
Das Studium solcher Probleme und im Besonderen Set-Packing und Covering-Probleme spielt eine bedeutende Rolle in der kombinatorischen Optimierung. Diese Probleme gehören zu den am häufigsten studierten mit einer weit entwickelten Theorie, die sich mit Begriffen wie perfekten, idealen, oder balancierten Matrizen, perfekten Graphen, der Theorie von blockierenden und anti-blockierenden Polyedern, Unabhängigkeitssystemen und semidefiniter Optimierung beschäftigt. Der Fokus dieses Abschnitts liegt in der (teilweisen) Beschreibung der zugehörigen Polyeder mittels Ungleichungen. Unter der Annahme, dass Relaxierungen von verschiedenen ganzzahligen Problemen auf Set-Packing oder Covering Probleme führen, kann das Wissen über diese Polyeder genutzt werden, um die Formulierung des Ausgangsproblems zu verschärfen.

Definition 3.26 *Sei $A \in \{0,1\}^{m \times n}$ eine 0/1-Matrix und $c \in \mathbb{R}^n$. Die ganzzahligen 0/1 Probleme*

$$\max\{\, c^T x : \; Ax \leq \mathbb{1},\, x \in \{0,1\}^n\}$$
$$\min\{\, c^T x : \; Ax \geq \mathbb{1},\, x \in \{0,1\}^n\}$$

nennen wir das Set-Packing *und das* Set-Covering Problem*, vgl. Beispiel 1.6.*

Jede Spalte j von A kann als der Inzidenzvektor einer Teilmenge F_j der Grundmenge $\{1, \ldots, m\}$ gesehen werden, d.h. es gilt $F_j := \{i \in \{1, \ldots, m\} : \; A_{ij} = 1\}$. Mit dieser Interpretation, besteht das Set-Packing Problem darin, eine Auswahl von Mengen aus $F_1, \ldots, F_n$ zu finden, die paarweise disjunkt und maximal bzgl. der Zielfunktion c sind. Analogerweise zielt das Covering-Problem auf das Auffinden einer Auswahl von Teilmengen, deren Vereinigung die Grundmenge überdeckt und die minimal bzgl. c ist.
Zulässige Mengen des Set-Packing Problems haben eine schöne graphentheoretische Interpretation. Wir führen einen Knoten für jeden Spaltenindex von A und eine Kante (i, j) zwischen zwei Knoten i und j ein, falls deren zugehörige Spalten einen gemeinsamen nichtverschwindenden Eintrag in derselben Zeile haben. Der resultierende Graph, den wir mit $G(A)$ bezeichnen, heißt (Spalten-) Konfliktgraph (engl. column intersection graph). Offensichtlich ist jeder zulässige 0/1-Vektor x, der die Ungleichung $Ax \leq \mathbb{1}$ erfüllt, der Inzidenzvektor einer *stabilen Menge* ($U \subseteq V$ ist eine stabile Menge, falls aus $i, j \in U$ die Aussage $(i, j) \notin E$ folgt) im Graphen $G(A)$. Umgekehrt ist der Inzidenzvektor einer stabilen Menge in $G(A)$ eine zulässige Lösung des Set-Packing Problems $Ax \leq \mathbb{1}$. Also ist das Studium der stabilen Mengen in Graphen dem Studium des Set-Packing Problems gleichwertig.
Wir betrachten nun eine 0/1-Matrix A und bezeichnen mit

$$P(A) = \operatorname{conv}\{x \in \{0,1\}^N : \; Ax \leq \mathbb{1}\}$$

das Set-Packing Polytop. Sei $G(A) = (V, E)$ der Konfliktgraph von A. Aus unseren bisherigen Überlegungen folgt, dass $P(A) = \text{conv}\{x \in \{0,1\}^n : x_i + x_j \leq 1, (i,j) \in E\}$ ist, wobei Letzteres eine Formulierung als ganzzahliges Problem des Problems der stabilen Mengen in G ist. Anders ausgedrückt, man erhält mit zwei Matrizen A und A' genau dasselbe Set-Packing Polytop, wenn deren zugehörige Konfliktgraphen übereinstimmen. Wir können also $P(A)$ betrachten mittels des Graphen G und bezeichnen nun das Set-Packing Polytop und das Stabile Mengen Polytop mit $P(G)$.
Zunächst ein paar einfache Beobachtungen bzgl. $P(G)$:

Bemerkung 3.27

(i) $P(G)$ *ist volldimensional.*

(ii) $P(G)$ *ist absteigend monoton, d.h.* $x \in P(G)$ *impliziert* $y \in P(G)$ *für alle* $0 \leq y \leq x$. *Alle nichttrivialen Facetten von* $P(G)$ *haben nichtnegative Koeffizienten.*

(iii) Die Nichtnegativitätsbedingungen $x_j \geq 0$ *induzieren Facetten von* $P(G)$, *siehe Übung.*

Aus Satz 2.22 wissen wir, dass die Kanten und die Nichtnegativitätsbedingungen genau dann genügen, um $P(G)$ zu beschreiben, wenn G bipartit ist. Nichtbipartite Graphen enthalten ungerade Kreise. Ungerade Kreise erzeugen neue gültige Ungleichungen, die nicht als Linearkombinationen von Kantenungleichungen erzeugt werden können.

Satz 3.28 *[31] Sei* $C \subseteq E$ *ein Kreis ungerader Kardinalität in* G. *Die* Ungerade-Kreis Ungleichung

$$\sum_{i \in V(C)} x_i \leq \frac{|V(C)| - 1}{2}$$

ist gültig für $P(G)$. *Die Ungleichung definiert genau dann eine Facette von* $P(\,(V(C), E(V(C))\,)$, *wenn* C *ein ungerades Loch ist, d.h. ein Kreis ohne Sehne.*

Beweis. Die Gültigkeit der Ungleichung ist klar.

Zur Charakterisierung der Facetten-Eigenschaft betrachte ein ungerades Loch C mit $V(C) = \{0, \ldots, k-1\}, k \in \mathbb{N}$, ungerade. Sei $b^T x \leq \beta$ eine facettendefinierende Ungleichung mit $F_a := \{x \in P : \sum_{i \in V(C)} x_i = \frac{|V(C)|-1}{2}\} \subseteq F_b := \{x \in P : b^T x = \beta\}$. Betrachte für ein $i \in V(C)$ die stabilen Mengen $S_1 = \{j : j = i+2, i+4, \ldots, i-3, i-1\}$ und $S_2 = \{j : j = i+2, i+4, \ldots, i-3, i\}$, wobei alle Indizes modulo k gerechnet sind. Dann gilt $\chi^{S_1}, \chi^{S_2} \in F_b$ und damit $0 = b^T\chi^{S_1} - b^T\chi^{S_2} = b_i - b_{i-1}$. Da i beliebig gewählt war, folgt $b_i = b_j$ für alle $i, j \in V(C)$. Damit ist $b^T x \leq \beta$ bis auf ein positives Vielfaches die Kreisungleichung.

Betrachte umgekehrt einen Kreis C mit Diagonalen. Dann enthält C ein ungerades Loch H unter Hinzunahme einer Kante $ij \in E(V(C)) \setminus C$. Betrachte paarweise verschiedene Knoten $i_l, j_l \in V(C) \setminus V(H)$ für $l = 1, \ldots, \frac{|V(C)|-|V(H)|}{2}$ mit $i_l j_l \in C$. Dann ist $\sum_{i \in V(C)} x_i \leq \frac{|V(C)|-1}{2}$ die Summe aus den gültigen Ungleichungen $\sum_{i \in V(H)} x_i \leq \frac{|V(H)|-1}{2}$ und $x_{i_l} + x_{j_l} \leq 1$ für $l = 1, \ldots, \frac{|V(C)|-|V(H)|}{2}$. Also kann es keine Facette induzieren. □

Ungerade-Kreis Ungleichungen können in polynomialer Zeit mittels des Algorithmus aus Lemma 9.1.11 in [21] separiert werden. Graphen $G = (V, E)$, für die $P(G)$ völlig durch die Kantenungleichungen $x_i + x_j \leq 1$ für $ij \in E$ und die Ungerader-Kreis Ungleichungen beschrieben wird, heißen t-perfekt, vgl. Chvátal [7]. Die Klasse der t-perfekten Graphen enthält eine Reihe von Graphen, wie z.B. serien-parallele und bipartite Graphen.
Eine weitere wichtige Klasse von gültigen Ungleichungen für das Stabile Mengen Polytop sind die Cliquenungleichungen.

Satz 3.29 *[14, 31] Sei $(C, E(C))$ eine Clique in G. Die Ungleichung*

$$\sum_{i \in C} x_i \leq 1$$

ist gültig für $P(G)$. Sie definiert genau dann eine Facette von $P(G)$, wenn $(C, E(C))$ maximal bzgl. Knoteninklusion ist.

Beweis. Übung. □

Graphen $G = (V, E)$, für die $P(G)$ vollständig durch die Cliquengleichungen beschrieben werden, heißen perfekt, eine Bezeichnungsweise, die auf Berge [4] zurückgeht.
Leider ist das Separierungsproblem für die Klasse der Cliquenungleichungen $\mathcal{NP}$-schwer, siehe Satz 9.2.9 in [21]. Überraschenderweise gibt es eine größere Klasse von Ungleichungen, die sog. orthonormalen Representations-Ungleichungen, die die Cliquenungleichungen umfassen und die in polynomialer Zeit separiert werden können. Neben den Kreis-, Cliquen und OR-Ungleichungen gibt es eine Menge anderer Ungleichungen, die für das stabile Mengen Polytop bekannt sind. Unter diesen findet man z.B. die Blüten-, ungerade Antiloch-, Räder-, Antinetz- und Netz-, Keil-Ungleichungen (engl. blossom, odd antihole, wheel, antiweb and web, wedge inequalities) und viele weitere. Auskunft darüber gibt [5] inklusive einer Diskussion deren Separierbarkeit.

3.2 Lagrange Relaxierungen

Im vorigen Abschnitt haben wir das gemischt ganzzahlige Problem mittels Relaxierung der Ganzzahligkeitsbedingungen zu lösen versucht und durch den Versuch, die Ganzzahligkeit der Lösung durch Hinzufügung von Schnittebenen zu

erreichen. In der Methode, die wir nun betrachten wollen, behalten wir die Ganzzahligkeitsbedingungen bei, relaxieren aber die Teile der Nebenbedingungsmatrix, die Schwierigkeiten erzeugt. Der gelöschte Teil wird wieder in das Problem eingeführt, indem er in die Zielfunktion mit Straftermen versehen wird. Betrachte wieder (3.1). Wir zerlegen A und b in zwei Teile $A = \binom{A_1}{A_2}$ und $b = \binom{b_1}{b_2}$, wobei $A_1 \in \mathbb{R}^{m_1 \times n}, A_2 \in \mathbb{R}^{m_2 \times n}, b_1 \in \mathbb{R}^{m_1}, b_2 \in \mathbb{R}^{m_2}$ mit $m_1 + m_2 = m$ ist. Dann wird (3.1) zu

$$\begin{aligned} \min \quad & c^T x \\ & A_1 x \leq b_1 \\ & A_2 x \leq b_2 \\ & x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p. \end{aligned} \tag{3.15}$$

Betrachte für ein festes $\lambda \in \mathbb{R}^{m_1}, \lambda \geq 0$ die folgende Funktion

$$\begin{aligned} L(\lambda) = \quad \min \quad & c^T x - \lambda^T (b_1 - A_1 x) \\ & x \in P^2, \end{aligned} \tag{3.16}$$

wobei $P^2 = \{x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p : A_2 x \leq b_2\}$ sei. (3.16) nennen wir die Lagrange Funktion. Offensichtlich ist (3.16) eine untere Schranke für (3.15), da für jede zulässige Lösung $\bar{x}$ von (3.15) gilt

$$c^T \bar{x} \geq c^T \bar{x} - \lambda^T (b_1 - A_1 \bar{x}) \geq \min_{x \in P^2} c^T x - \lambda^T (b_1 - A_1 x) = L(\lambda)$$

Da dies für jedes $\lambda \geq 0$ gilt, erhalten wir mit

$$\max_{\lambda \geq 0} L(\lambda) \tag{3.17}$$

eine untere Schranke für (3.1). (3.17) heißt Lagrange Relaxierung. Häufig wird 3.17 das "Lagrange Dual" bezeichnet und der gesamte Ansatz Lagrange Relaxierung. Sei λ^* eine optimale Lösung von (3.17). Es bleiben die Fragen, wie gut ist $L(\lambda^*)$ und wie kann λ^* berechnet werden. Eine Antwort auf die erste Frage erhalten wir durch folgenden Satz.

Satz 3.33 $L(\lambda^*) = \min\{c^T x : A_1 x \leq b_1, x \in conv(P^2)\}$.

Beweis. Es gilt

$$\begin{aligned} L(\lambda) = & \min_{x \in P^2} c^T x - \lambda^T (b_1 - A_1 x) \\ & \min_{x \in \mathrm{conv}\,(P^2)} c^T x - \lambda^T (b_1 - A_1 x) \end{aligned}$$

Und damit

$$\begin{aligned} L(\lambda^*) &= \max_{\lambda \geq 0} L(\lambda) \\ &= \max_{\lambda \geq 0} \min_{x \in \text{conv}\,(P^2)} c^T x - \lambda^T (b_1 - A_1 x). \end{aligned}$$

Falls $P^2 = \emptyset$ ist das innere Minimum $+\infty$ für alle λ und damit auch $L(\lambda^*)$. Andernfalls existieren Vektoren $v_1, \ldots, v_k$ und $e_1, \ldots, e_l$, so dass $\text{conv}\,(P^2) = \text{conv}\,(\{v_1, \ldots, v_k\}) + \text{cone}\,(\{e_1, \ldots, e_l\})$. Damit gilt

$$\min_{x \in \text{conv}\,(P^2)} c^T x - \lambda^T (b_1 - A_1 x) = \begin{cases} -\infty & \text{falls } (c^T + \lambda^T A_1) e_i < 0 \\ & \text{für ein } i \in \{1, \ldots, l\} \\ c^T v_j - \lambda^T (b_1 - A_1 v_j) & \text{für ein } j \in \{1, \ldots, k\} \\ & \text{andernfalls.} \end{cases}$$

Also gilt

$$\begin{aligned} L(\lambda^*) = \max_{\lambda \geq 0} \min_{j \in \{1,\ldots,k\}} & c^T v_j - \lambda^T (b_1 - A_1 v_j) \\ & (c^T + \lambda^T A_1) e_i \geq 0 \text{ für } i = 1, \ldots, l. \end{aligned} \tag{3.18}$$

Letzteres ist äquivalent zu

$$\begin{aligned} L(\lambda^*) = \quad & \max_{\lambda \geq 0, \eta \in \mathbb{R}} \ \eta \\ \eta + \lambda^T (b_1 - A_1 v_j) &\leq c^T v_j \quad \text{für } j = 1, \ldots, k \\ -\lambda^T A_1 e_i &\leq c^T e_i \quad \text{für } i = 1, \ldots, l. \end{aligned}$$

Mit dem Dualitätssatz der Linearen Programmierung erhalten wir schließlich

$$\begin{aligned} L(\lambda^*) = \min & \, c^T (\textstyle\sum_{j=1}^k \alpha_j v_j + \sum_{i=1}^l \beta_i e_i) \\ & \textstyle\sum_{j=1}^k \alpha_j = 1 \\ & -A_1 (\textstyle\sum_{j=1}^k \alpha_j v_j + \sum_{i=1}^l \beta_i e_i) \geq -b_1 (\sum_{j=1}^k \alpha_j) \\ & \alpha_j \geq 0 \text{ für } j = 1, \ldots, k \\ & \beta_i \geq 0 \text{ für } i = 1, \ldots, l \\ = \min & \{c^T x : \ A_1 x \leq b_1, x \in \text{conv}\,(P^2)\}. \end{aligned}$$

□

Wegen

$$\begin{aligned} \{x \in \mathbb{R}^n : \ Ax \leq b\} &\supseteq \{x \in \mathbb{R}^n : \ A_1 x \leq b_1, \ x \in \text{conv}\,(P^2)\} \\ &\supseteq \text{conv}\,\{x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p : \ Ax \leq b\} \end{aligned}$$

erhalten wir aus Satz 3.33

Folgerung 3.34

$$z_{LP} \leq L(\lambda^*) \leq z_{IP},$$

wobei z_{LP} der optimale Lösungswert der LP-Relaxierung und z_{IP} der optimale ganzzahlige Lösungswert ist.

Es bleibt zu überlegen, auf welche Art $L(\lambda^*)$ berechnet werden kann. Aus theoretischer Sicht kann gezeigt werden, dass unter der Ausnutzung der Äquivalenz von Separieren und Optimieren $L(\lambda^*)$ in polynomialer Zeit berechnet werden kann, wenn $\min\{\tilde{c}^T x : x \in \text{conv}\,(P^2)\}$ in polynomialer Zeit für jede Zielfunktion $\tilde{c}$ berechnet werden kann, siehe [35]. Wir werden darauf später noch einmal zurückkommen.

Zur Berechnung von $L(\lambda^*)$ werden häufig Subgradientenverfahren verwendet. Dazu folgende Überlegungen:

Satz 3.35 *Die Funktion $L(\lambda)$ ist stückweise linear und konkav auf dem Definitionsbereich, über dem sie beschränkt ist.*

Beweis.
Zunächst beobachten wir, dass $L(\lambda)$ genau dann endlich ist, wenn λ in dem Polyeder $\{\hat{\lambda} \in \mathbb{R}^{m_1}_+ : -\hat{\lambda}^T A_1 e_i \leq c^T e_i,\ i = 1, \ldots, l\}$ liegt, siehe (3.18). In diesem Polyeder gilt $L(\lambda) = \min_{j\in\{1,\ldots,k\}} c^T v_j - \lambda^T(b_1 - A_1 v_j)$, d.h. $L(\lambda)$ ist das Minimum über eine endliche Anzahl affiner Funktionen.
Bleibt zu zeigen, dass die Funktion konkav ist. Betrachte dazu $\lambda^3 = \alpha\lambda^1 + (1-\alpha)\lambda^2$ für $\lambda^1, \lambda^2, \lambda^3 \geq 0$. Zu zeigen ist, $L(\lambda^3) \geq \alpha L(\lambda^1) + (1-\alpha)L(\lambda^2)$. Seien v_{j_t} die zugehörigen Optimallösungen mit $L(\lambda^t) = c^T v_{j_t} + (\lambda^t)^T(b_1 - A_1 v_{j_t})$ für $t = 1, 2, 3$. Dann gilt

$$\begin{aligned}
L(\lambda^3) &= c^T v_{j_3} - (\lambda^3)^T(b_1 - A_1 v_{j_3}) \\
&= c^T v_{j_3} - (\alpha\lambda^1 + (1-\alpha)\lambda^2)^T(b_1 - A_1 v_{j_3}) \\
&= \alpha(c^T v_{j_3} - (\lambda^1)^T(b_1 - A_1 v_{j_3})) + (1-\alpha)(c^T v_{j_3} - (\lambda^2)^T(b_1 - A_1 v_{j_3})) \\
&\geq \alpha(c^T v_{j_1} - (\lambda^1)^T(b_1 - A_1 v_{j_1})) + (1-\alpha)(c^T v_{j_2} - (\lambda^2)^T(b_1 - A_1 v_{j_2})) \\
&= \alpha L(\lambda^1) + (1-\alpha)L(\lambda^2).
\end{aligned}$$

□

Definition 3.36 *Sei $L : \mathbb{R}^{m_1} \mapsto \mathbb{R}$ eine konkave Funktion. Dann heißt ein Vektor $u \in \mathbb{R}^{m_1}$* Subgradient *an L im Punkt λ_0, falls*

$$L(\lambda) - L(\lambda_0) \leq u^T(\lambda - \lambda_0)$$

für alle $\lambda \in \mathbb{R}^{m_1}$.

Satz 3.37

(a) *Betrachte* $\lambda^0 \in \mathbb{R}^{m_1}_+$ *und eine Optimallösung* x^0 *für (3.16). Dann ist* $g^0 = A_1x^0 - b_1$ *ein Subgradient an* L *in* λ^0.

(b) λ^* *maximiert* L *genau dann, wenn* 0 *ein Subgradient an* L *in* λ^* *ist.*

Beweis.
Zu (a): $L(\lambda) - L(\lambda^0) = c^Tx^\lambda - \lambda^T(b_1 - A_1x^\lambda) - (c^Tx^0 - (\lambda^0)^T(b_1 - A_1x^0)) \leq c^Tx^0 - \lambda^T(b_1 - A_1x^0) - (c^Tx^0 - (\lambda^0)^T(b_1 - A_1x^0)) = (g^0)^T(\lambda - \lambda^0)$.
Zu (b): Ist 0 ein Subgradient in λ^*, so gilt $L(\lambda) - L(\lambda^*) \leq 0^T(\lambda - \lambda^*) = 0$ für alle $\lambda \in \mathbb{R}^{m_1}$. Also maximiert λ^* die Funktion L. Umgekehrt, falls $L(\lambda^*)$ maximal ist, gilt $L(\lambda^*) \geq L(\lambda)$ für alle $\lambda \in \mathbb{R}^{m_1}$, und damit $L(\lambda) - L(\lambda^*) \leq 0 = 0^T(\lambda - \lambda^*)$. Folglich ist 0 ein Subgradient. □

Also gilt für λ^* die Aussage $(g^0)^T(\lambda^* - \lambda^0) \geq L(\lambda^*) - L(\lambda^0) \geq 0$. Daher können wir, um λ^* zu finden, mit einem λ^0 beginnen, dann berechnen wir $x^0 = \text{argmin}\{c^Tx - (\lambda^0)^T(b_1 - A_1x) : x \in P^2\}$ und bestimmen iterativ $\lambda^0, \lambda^1, \lambda^2, \ldots$ indem wir setzen $\lambda^{k+1} = \lambda^k + \mu^k g^k$, wobei μ^k eine noch zu bestimmende Schrittlänge ist. Diese iterative Methode ist die Essenz der Subgradientenmethode.

Algorithmus 3.38 *(Subgradientenmethode)*
Input: *Konkave Funktion* $L : \mathbb{R}^n \mapsto \mathbb{R}$.
Output: $\max\{L(\lambda) : \lambda \in \mathbb{R}^n_+\}$.

(1) Wähle $\lambda_0 \in \mathbb{R}^n$ *beliebig. Setze* $k = 0$.
(2) Berechne $L(\lambda_k)$. *Sei* x^k *zugehörige Optimallösung.*
(3) Ist ein bestimmtes Stop-Kriterium erfüllt, **Stop** *(gib* λ_k *und* x^k *aus).*
(4) Wähle ein neues λ^{k+1} *durch*

$$\lambda^{k+1} = \lambda^k + \mu^k g^k$$

wobei μ^k *eine zu spezifizierende Schrittlänge ist.*
(5) Setze $k = k + 1$ *und gehe nach (2).*

Satz 3.39 *[34]*
Falls die konkave Funktion $L : \mathbb{R}^{m_1} \mapsto \mathbb{R}$ *nach oben beschränkt ist und die Folge* (μ^k) *der Schrittlängen* $\lim_{k\to\infty} \mu^k = 0$ *und* $\sum_{k=0}^{\infty} \mu^k = \infty$ *erfüllt, so gilt*

$$\lim_{k\to\infty} L(\lambda^k) = L(\lambda^*).$$

Natürlich hängt die Qualität der Lagrange Relaxierung sehr davon ab, welche Menge an Bedingungen relaxiert wird. Einerseits müssen wir (3.16) für verschiedene Werte von λ bestimmen und damit ist es notwendig, einen Wert von $L(\lambda)$ schnell zu berechnen. Es ist einerseits also wünschenswert, so viele (komplizierte)

Bedingungen als möglich zu relaxieren. Andererseits, je mehr Bedingungen relaxiert werden, desto schlechter wird die Schranke $L(\lambda^*)$ werden. Also muss man einen Kompromiss zwischen diesen beiden sich widersprechenden Zielen finden. Im Folgenden geben wir einige Anwendungen, bei denen diese Methode erfolgreich angewendet werden konnte und eine gute Balance zwischen den beiden gegensätzlichen Zielen gefundenen werden kann.

Anwendungen

Eine Anwendung ist das Problem des Handlungsreisenden. Durch Relaxierung der Gradbedingungen in der IP-Formulierung für dieses Problem bleibt das Problem des Auffindens eines aufspannenden Baumes erhalten, das leicht mit dem Greedy-Algorithmus gelöst werden kann. Ein Hauptvorteil dieser TSP-Relaxierung ist, dass für die Berechnung von (3.16) kombinatorische Algorithmen vorhanden sind und kein allgemeines LP oder IP Lösungsverfahren angewendet werden muss.

Lagrange Relaxierungen werden sehr oft benutzt, wenn die zugrundeliegenden LP's von (3.1) einfach zu groß sind, um direkt gelöst werden zu können und sogar die relaxierten Probleme in (3.16) sind noch sehr groß. Oft kann die Relaxierung in der Weise getan werden, dass die Berechnung von (3.16) kombinatorisch geschehen kann. Andere Beispiele sind Multicommodity Flow-Probleme, die z.B. in der Fahrzeugumlaufplanung oder in Zerlegungen von stochastischen gemischt ganzzahligen Problemen auftreten. In der Tat gehören die letzten beiden Anwendungen zu einer Klasse von Problemen, wobei die zugrundeliegende Matrix (nahezu) Blockdiagonalform hat, siehe Bild 3.1. Relaxieren wir die Kopplungsbedingungen in einer Lagrange Relaxierung, so zerfällt die verbleibende Matrix in k unabhängige Blöcke. Also ist ein Wert $L(\lambda)$ die Summe von k unabhängigen Termen, die getrennt bestimmt werden können. Oft stellt jeder einzelne Block A_i ein Netzwerk Flussproblem, ein Rucksackproblem oder dergleichen dar und kann daher unter Benutzung spezieller kombinatorischer Algorithmen gelöst werden.

3.3 Dekompositionsmethoden

3.3.1 Dantzig-Wolfe Dekomposition

Die Idee von Dekompositionsmethoden besteht darin, eine Menge von Bedingungen (Variablen) aus dem Problem herauszunehmen und diese auf einer übergeordneten Ebene (oft Masterproblem genannt) zu betrachten. Das resultierende untergeordnete Problem kann oft effizienter gelöst werden. Dekompositionsmethoden arbeiten nun abwechselnd auf dem Master- und dem Subproblem und tauschen iterativ Informationen aus, um das Ausgangsproblem optimal zu lösen. In diesem Abschnitt betrachten wir zwei bekannte Beispiele dieses Ansatzes, die Dantzig-Wolfe Dekomposition und Benders' Dekomposition. Wir werden erken-

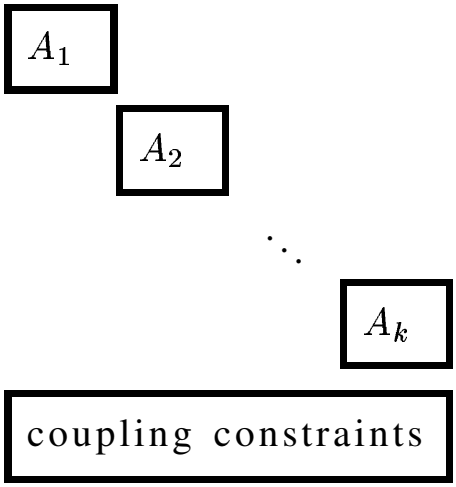

Abbildung 3.1: Matrix in begrenzter Blockdiagonalform

nen, dass wie im Fall der Lagrange Relaxierung diese Methoden ebenso einen Teil der Nebenbedingungsmatrix weglassen. Aber anstelle diesen Teil wieder in die Zielfunktion einzuführen, wird dieser nun umformuliert und wieder in das System der Nebenbedingungen eingeführt.
Wir beginnen mit der Dantzig-Wolfe Dekomposition [9] und betrachten wieder (3.15), wobei wir zunächst annehmen, dass $p = 0$, d.h. wir haben ein LP. Betrachten wir das Polyeder $P^2 = \{x \in \mathbb{R}^n : A_2 x \leq b_2\}$. Wir wissen aus Kapitel 1, dass es Vektoren $v_1, \ldots, v_k$ und $e_1, \ldots, e_l$ gibt, so dass $P^2 = \text{conv}(\{v_1, \ldots, v_k\}) + \text{cone}(\{e_1, \ldots, e_l\})$. In anderen Worten, $x \in P^2$ kann in der Form

$$x = \sum_{i=1}^{k} \lambda_i v_i + \sum_{j=1}^{l} \mu_j e_j \tag{3.19}$$

mit $\lambda_1, \ldots, \lambda_k \geq 0, \sum_{i=1}^{k} \lambda_i = 1$ und $\mu_1, \ldots, \mu_l \geq 0$ geschrieben werden. Ersetzen wir x aus (3.19), so können wir (3.15) schreiben als

$$\begin{aligned} \min \quad & c^T(\sum_{i=1}^{k} \lambda_i v_i + \sum_{j=1}^{l} \mu_j e_j) \\ & A_1(\sum_{i=1}^{k} \lambda_i v_i + \sum_{j=1}^{l} \mu_j e_j) \leq b_1 \\ & \sum_{i=1}^{k} \lambda_i = 1 \\ & \lambda \in \mathbb{R}^k_+, \mu \in \mathbb{R}^l_+, \end{aligned}$$

was gleichwertig ist zu

$$\begin{aligned} \min \quad & \sum_{i=1}^{k} (c^T v_i)\, \lambda_i + \sum_{j=1}^{l} (c^T e_j)\, \mu_j \\ & \sum_{i=1}^{k} (A_1 v_i)\, \lambda_i + \sum_{j=1}^{l} (A_1 e_j)\, \mu_j \le b_1 \\ & \sum_{i=1}^{k} \lambda_i = 1 \\ & \lambda \in \mathbb{R}^k_+, \mu \in \mathbb{R}^l_+. \end{aligned} \tag{3.20}$$

(3.20) nennt man das Masterproblem von (3.15). Beim Vergleich der Formulierungen (3.15) und (3.20) erkennen wir, dass wir die Zahl der Nebenbedingungen von m auf m_1, reduziert haben, aber wir haben nun $k + l$ Variablen anstelle von n. $k + l$ kann im Vergleich zu n groß sein, ja sogar exponentiell (betrachte z.B. den Einheitswürfel im $\mathbb{R}^n$ mit $2n$ Nebenbedingungen und 2^n Ecken), so dass auf den ersten Blick kein Vorteil in der Anwendung von (3.20) erkennbar ist. Dennoch können wir die Simplexmethode für die Lösung von (3.20) verwenden. Wir kürzen (3.20) ab durch $\min\{w^T\eta : D\eta = d, \eta \ge 0\}$ mit $D \in \mathbb{R}^{(m_1+1)\times(k+l)}, d \in \mathbb{R}^{m_1+1}$. Man erinnere sich, dass die Simplexmethode mit einer (zulässigen) Basis $B \subseteq \{1, \ldots, k+l\}$, $|B| = m_1+1$, mit D_B regulär und der zugehörigen (zulässigen) Lösung $\eta^*_B = D_B^{-1} d$ und $\eta^*_N = 0$ mit $N = \{1, \ldots, k+l\}\backslash B$ startet. Beachte, dass $D_B \in \mathbb{R}^{(m_1+1)\times(m_1+1)}$ (viel) kleiner als eine Basis für das ursprüngliche System (3.15) ist und dass nur ein Teil der Variablen ($m_1 + 1$ von $k + l$) nichtverschwinden können. Zusätzlich gilt, dass auf dem Weg zu einer optimalen Lösung die einzige Operation in der Simplexmethode, die alle Spalten benutzt, der Schritt Pricing ist, bei dem geprüft wird, ob die reduzierten Kosten $w_N - \tilde{y}^T D_N$ nichtnegativ mit $\tilde{y} \le 0$ als Lösung von $y^T D_B = w_B$ ist. Die Nichtnegativität der reduzierten Kosten kann mittels des folgenden linearen Programms geprüft werden:

$$\begin{aligned} \min \quad & (c^T - \bar{y}^T A_1)x \\ & A_2 x \le b_2 \\ & x \in \mathbb{R}^n, \end{aligned} \tag{3.21}$$

Dabei sind $\bar{y}$ die ersten m_1 Komponenten der Lösung von $\tilde{y}$. Die folgenden Fälle können auftreten:

(i) (3.21) hat eine optimale Lösung $\tilde{x}$ mit $(c^T - \bar{y}^T A_1)\tilde{x} < \tilde{y}_{m_1+1}$.

In diesem Fall ist $\tilde{x}$ einer der Vektoren $v_i, i \in \{1, \ldots, k\}$ mit zugehörigen reduzierten Kosten

$$w_i - \tilde{y}^T D_{\cdot i} = c^T v_i - \tilde{y}^T \begin{pmatrix} A_1 v_i \\ 1 \end{pmatrix} = c^T v_i - \bar{y}^T A_1 v_i - \tilde{y}_{m_1+1} < 0.$$

In anderen Worten: $\binom{A_1 v_i}{1}$ ist die eintretende Spalte im Simplexalgorithmus.

(ii) (3.21) ist unbeschränkt.

Hier erhalten wir einen zulässigen Extremstrahl e^* mit $(c^T - \bar{y}^T A_1)e^* < 0$. e^* ist einer der Vektoren e_j, $j \in \{1, \ldots, l\}$. Wir erhalten eine Spalte $\binom{A_1 e_j}{0}$ mit reduzierten Kosten

$$w_{k+j} - D_{\cdot(k+j)} = c^T e_j - \tilde{y}^T \begin{pmatrix} A_1 e_j \\ 0 \end{pmatrix} = c^T e_j - \bar{y}^T (A_1 e_j) < 0.$$

Also ist $\binom{A_1 e_j}{0}$ die eintretende Spalte.

(iii) (3.21) hat eine optimale Lösung $\tilde{x}$ mit $(c^T - \bar{y}^T A_1)^T \tilde{x} \geq \tilde{y}_{m_1+1}$.

In diesem Fall erhalten wir mit denselben Argumenten wie in (i) und (ii), dass $w_i - \tilde{y}^T D_{\cdot i} \geq 0$ für alle $i = 1, \ldots, k+l$, was zeigt, dass x^* eine optimale Lösung des Masterproblems (3.20) ist.

Man beachte, dass das ganze Problem (3.15) in zwei Teilprobleme zerlegt wird, d.h. in (3.20) und (3.21) und dieser Ansatz arbeitet iterativ auf der höheren Ebene (3.20) und auf der niedrigeren Ebene (3.21). Das Verfahren beginnt mit einer zulässigen Lösung für (3.20) und erzeugt neue erfolgversprechende Spalten nach Bedarf durch Lösung von (3.21). Solche Verfahren werden gewöhnlich spaltenerzeugende oder verzögerte spaltenerzeugende Algorithmen (engl. delayed column generation algorithms) genannt.
Dieser Ansatz kann ebenso auf allgemeine IP's mit einiger Vorsicht übertragen werden. In diesem Fall verändert sich das Problem (3.21) von einem linearen zu einem ganzzahligen linearen Problem. Zusätzlich müssen wir sicherstellen, dass in (3.19) alle zulässigen ganzzahligen Lösungen x von (3.15) erzeugt werden können durch (ganzzahlige) Linearkombinationen der Vektoren $v_1, \ldots, v_k$ und $e_1, \ldots, e_l$ mit

$$\text{conv}\{x \in \mathbb{Z}^n : Ax \leq b\} = \text{conv}(v_1, \ldots, v_k) + \text{cone}(e_1, \ldots, e_l).$$

Es genügt nicht zu verlangen, dass λ und μ ganzzahlig sein müssen. Betrachte als Gegenbeispiel $A_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $b_1 = \binom{1.5}{1.5}$ und $A_2 = (1,1), b_2 = 2$ sowie das Problem

$$\max\{x_1 + x_2 : A_1 x \leq b_1, A_2 x \leq b_2, x \in \{0,1,2\}^2\}$$

Dann ist $P^2 = \text{conv}(\{\binom{0}{0}, \binom{2}{0}, \binom{0}{2}\})$, siehe Abbildung 3.2, aber die optimale Lösung $\binom{1}{1}$ des ganzzahligen Problems ist keine ganzzahlige Linearkombination der Ecken von P^2. Sind jedoch alle Variablen 0/1, so tritt diese Schwierigkeit nicht auf, da jede 0/1-Lösung der LP-Relaxierung eines binären MIP immer eine Ecke dieses Polyeders ist (Übung). In der Tat werden spaltenerzeugende Algorithmen nicht nur für die Lösung von großen linearen Problemen benutzt, sondern insbesondere für große binäre ganzzahlige Probleme.

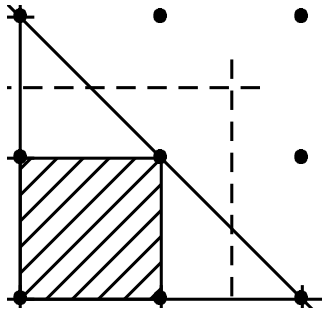

Abbildung 3.2: Erweiterung der Dantzig-Wolfe Dekomposition auf IP's

Natürlich ist die Dantzig-Wolfe Zerlegung für lineare oder binäre ganzzahlige Probleme nur eine Art von spaltenerzeugenden Algorithmen. Andere Autoren lösen das untergeordnete Problem nicht mittels allgemeiner Techniken für LP oder IP Probleme sondern durch kombinatorische oder explizite enumerative Algorithmen. Weiterhin werden die Probleme oft nicht über (3.15) modelliert, sondern direkt wie in (3.20). Dies ist z.B. der Fall, wenn die Menge der zulässigen Lösungen eine komplexe Beschreibung durch lineare Ungleichungen hat, aber diese Bedingungen leicht in ein Enumerationsschema eingebaut werden können.

3.3.2 Benders' Dekomposition

Abschließend wollen wir Benders' Dekomposition [3] betrachten. Benders' Dekomposition eliminiert auch einen Teil der Nebenbedingungsmatrix, aber anders als bei der Dantzig-Wolfe Dekomposition, bei der wir einen Teil der Nebenbedingungen löschen und diese wieder mittels Spaltenerzeugung erneut einfügen, löschen wir nun einen Teil der Variablen und führen diese wieder mittels Schnittebenen ein. Aus dieser Sicht ist Benders' Dekomposition gleich der Dantzig-Wolfe Dekomposition angewendet auf das duale Problem, was wir in Abschnitt 3.4 sehen werden. Betrachte wieder (3.1) und schreibe es in der Form

$$\begin{aligned} \min \quad & c_1^T x_1 + c_2^T x_2 \\ & A_1 x_1 + A_2 x_2 \leq b \\ & x_1 \in \mathbb{R}^{n_1}, x_2 \in \mathbb{R}^{n_2}, \end{aligned} \tag{3.22}$$

wobei $A = [A_1, A_2] \in \mathbb{R}^{m \times n}, A_1 \in \mathbb{R}^{m \times n_1}, A_2 \in \mathbb{R}^{m \times n_2}$, $c_1, x_1 \in \mathbb{R}^{n_1}$, $c_2, x_2 \in \mathbb{R}^{n_2}$ mit $n_1 + n_2 = n$ gilt. Beachte, dass wir zur Einfachheit der Darstellung den kontinuierlichen Fall angenommen haben. Wir werden jedoch sehen, dass alle Ergebnisse auch für den Fall $x_1 \in \mathbb{Z}^{n_1}$ gültig sind. Wir wollen die Variablen x_2 entfernen. Diese Variablen halten (3.22) davon ab, ein reines ganzzahliges Programm zu sein, falls $x_1 \in \mathbb{Z}^{n_1}$. Auch im Fall eines LP's können sie der Grund für einige Schwierigkeiten sein, siehe beispielsweise die Anwendungen aus Stochasti-

schen Programmierung. Ein bekannter Ansatz zur Entfernung von Variablen ist die Projektion.
Um Projektion anwenden zu können, müssen wir (3.22) umformulieren zu

$$\begin{aligned} \min \quad & z \\ & -z + c_1^T x_1 + c_2^T x_2 \leq 0 \\ & A_1 x_1 + A_2 x_2 \leq b \\ & z \in \mathbb{R},\ x_1 \in \mathbb{R}^{n_1},\ x_2 \in \mathbb{R}^{n_2}, \end{aligned} \tag{3.23}$$

Nun ist (3.23) äquivalent zu (vgl. Fourier-Motzkin Elimination aus Kapitel 1)

$$\begin{aligned} \min \quad & z \\ & -uz + uc_1^T x_1 + v^T A_1 x_1 \leq v^T b \\ & z \in \mathbb{R},\ x_1 \in \mathbb{R}^{n_1}, \\ & \tbinom{u}{v} \in C, \end{aligned} \tag{3.24}$$

mit

$$C = \{\tbinom{u}{v} \in \mathbb{R}^{m+1} :\ v^T A_2 + uc_2^T = 0,\ u \geq 0, v \geq 0\}.$$

C ist ein spitzer polyedrischer Kegel, es gibt also Vektoren $\binom{\bar{u}_1}{\bar{v}_1}, \ldots, \binom{\bar{u}_s}{\bar{v}_s}$, so dass gilt $C = \operatorname{cone}(\{\binom{\bar{u}_1}{\bar{v}_1}, \ldots, \binom{\bar{u}_s}{\bar{v}_s}\})$. Diese Extremalstrahlen können so reskaliert werden, dass $\bar{u}_i$ zu null oder eins wird. Es folgt $C = \operatorname{cone}(\{\binom{0}{v_k} : k \in K\}) + \operatorname{cone}(\{\binom{1}{v_j} : j \in J\})$ mit $K \cup J = \{1, \ldots, s\}$ und $K \cap J = \emptyset$, vgl. Kapitel 1. Mit dieser Beschreibung von C kann (3.24) dargestellt werden als

$$\begin{aligned} \min \quad & z \\ & -z \leq c_1^T x_1 + v_j^T (b - A_1 x_1) \quad \text{für alle } j \in J, \\ & 0 \leq \quad v_k^T (b - A_1 x_1) \quad \text{für alle } k \in K, \\ & z \in \mathbb{R}, x_1 \in \mathbb{R}^{n_1}. \end{aligned} \tag{3.25}$$

(3.25) nennt man Benders' Masterproblem. Benders' Masterproblem hat nur $n_1 + 1$ Variablen anstelle von $n_1 + n_2$ Variablen in (3.22) oder im Fall $x_1 \in \mathbb{Z}^{n_1}$ haben wir das gemischt ganzzahlige Programm (3.22) in ein i.W. rein ganzzahliges Programm (3.25) mit einer zusätzlichen kontinuierlichen Variablen z umgeschrieben. Dennoch enthält (3.25) eine große Anzahl an Bedingungen, i.A. exponentiell viele in n. Um dieses Problem zu umgehen, lösen wir Benders' Masterproblem mittels Schnittebenenverfahren, siehe Abschnitt 3.1. Wir beginnen mit einer kleinen Teilmenge von Extremstrahlen von C (möglicherweise mit der leeren Menge) und optimieren (3.25) einfach über dieser Teilmenge. Wir erhalten eine optimale Lösung

x^*, z^* des relaxierten Problems und wir müssen prüfen, ob diese Lösung alle anderen Ungleichungen in (3.25) erfüllt. Dies kann mittels des folgenden linearen Programms getan werden, vgl. (3.24):

$$\begin{aligned} \min \quad & v^T(b - A_1 x_1^*) + u(z^* - c_1^T x_1^*) \\ & \binom{u}{v} \in C. \end{aligned} \tag{3.26}$$

(3.26) heißt Benders' Teilproblem. Es ist zulässig, da $\binom{0}{0} \in C$ und (3.26) hat eine optimale Lösung mit Wert Null oder ist unbeschränkt. Im ersten Fall erfüllt x_1^*, z^* alle Ungleichungen in (3.25) und wir haben (3.25) gelöst und damit (3.22). Im anderen Fall erhalten wir einen Extremstrahl $\binom{u^*}{v^*}$ aus (3.26) mit $(v^*)^T(b - A_1 x_1^*) + u^*(z^* - c_1^T x_1^*) < 0$, der nach Reskalierung einen Schnitt für (3.25) erzeugt, der von x_1^*, z^* verletzt wird. Wir fügen diesen Schnitt zu Benders' Masterproblem (3.25) hinzu und iterieren.

3.4 Verbindungen zwischen diesen Ansätzen

Auf den ersten Blick scheinen Lagrange Relaxierung, Dantzig-Wolfe- und Benders Dekomposition vollständig verschiedene Relaxierungsansätze zu sein. Sie sind allerdings stark miteinander verbunden, wie wir im Folgenden kurz zeigen wollen. Betrachte noch einmal (3.21), das für ein festes $\bar{y} \leq 0$ geschrieben werden kann als

$$\begin{aligned} \min \quad & (c^T - \bar{y}^T A_1)x \\ & x \in P^2 \end{aligned} \quad = \quad \begin{aligned} \min \quad & c^T x + \bar{y}^T(b_1 - A_1 x) - \bar{y}^T b_1 \\ & x \in P^2 \end{aligned}$$

$$= \quad L(-\bar{y}) - \bar{y}^T b,$$

d.h. (3.16) und (3.21) stellen dieselben Probleme bis auf die Konstante $-\bar{y}^T b$ dar. Weiterhin kann man durch Ersetzen von P^2 durch $\operatorname{conv}(\{v_1, \ldots, v_k\}) + \operatorname{cone}(\{e_1, \ldots, e_l\})$ zeigen, dass (3.20) mit der rechten Seite in Satz 3.33 übereinstimmt und somit mit $L(\lambda^*)$. In anderen Worten, sowohl Dantzig-Wolfe und Lagrange Relaxierung berechnen dieselbe Schranke. Die einzigen Unterschiede sind, dass für den Update die dualen Variablen, d.h. λ in der Lagrange Relaxierung und $\bar{y}$ in Dantzig-Wolfe, im ersten Fall Subgradientmethoden wohingegen im zweiten Fall LP-Techniken angewendet werden. Andere Möglichkeiten zur Berechnung von λ^* ergeben sich aus der Bündelmethode, die auf quadratischer Optimierung [23] beruht und aus der Analytic-Center Schnittebenenmethode, die auf einem Innere Punkte Algorithmus beruht [17].

Analogerweise ist Benders' Dekomposition nichts anderes als Dantzig-Wolfe angewendet auf das Duale von (3.22). Um dies zu sehen, betrachte dessen duales

lineare Programm

$$\begin{array}{rl} \max & -y^T b \\ & -y^T A_1 = c_1^T \\ & -y^T A_2 = c_2^T \\ & y \quad \geq 0. \end{array} \tag{3.27}$$

Nun schreibe $\bar{P}^2 = \{y \in \mathbb{R}^{n_2} : y^T A_2 = -c_2^T, y \geq 0\}$ als $\bar{P}^2 = \text{conv}(\{v_j : j \in J\}) + \text{cone}(\{v_k : k \in K\})$, wobei K, J und v_l, $l \in K \cup J$ genau die Werte aus (3.25) sind (beachte, $\text{hog}(\bar{P}^2) = C$, d.h. C ist die Homogenisierung von $\bar{P}^2$)[1] und schreibe (3.27) als

$$\begin{array}{rl} \max & \displaystyle\sum_{j\in J}(-v_j^T b)\,\lambda_j + \sum_{k\in K}(-v_k^T b)\,\mu_k \\ & \displaystyle\sum_{j\in J}(-v_j^T A_1)\,\lambda_j + \sum_{k\in K}(-v_k^T A_1)\,\mu_k = c_1^T \\ & \displaystyle\sum_{j\in J}\lambda_j \qquad\qquad\qquad\qquad\qquad\qquad = 1 \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad \lambda \in \mathbb{R}^J_+, \mu \in \mathbb{R}^K_+. \end{array} \tag{3.28}$$

Wir folgern nun mit den Ergebnissen aus Abschnitt 3.3.1, dass (3.28) das Masterproblem von (3.27) ist. Dualisierung von (3.28) ergibt nun

$$\begin{array}{rlll} \min & c_1^T x_1 + z & & \\ & v_j^T(b - A_1 x_1) & \geq \quad -z & \forall\ j \in J \\ & v_k^T(b - A_1 x_1) & \geq \quad 0 & \forall\ k \in K, \end{array}$$

was gleichwertig ist zu (3.25), also zu Benders' Masterproblem von (3.22). In anderen Worten: Benders' und Dantzig-Wolfe Dekomposition ergeben im Falle von linearen Programmen dieselbe Schranke, die nach den eingangs gemachten Überlegungen gleich dem Wert der Lagrange Relaxierung (3.17) ist.

[1] Für $S \subseteq \mathbb{K}^n$ heißt $\text{hog}(S) = \left\{\binom{x}{1} \in \mathbb{K}^{n+1} \mid x \in S\right\}^{\circ\circ}$ Homogenisierung von S. Für Polyeder $P = P(A,b) = \text{conv}(V) + \text{cone}(E)$ gilt $\text{hog}(P) = P(B,0) = \text{cone}\left(\left\{\binom{v}{1} \mid v \in V\right\}\right) + \text{cone}\left(\left\{\binom{e}{0} \mid e \in E\right\}\right)$, wobei $B = \begin{pmatrix} A & -b \\ 0 & -1 \end{pmatrix}$.

Kapitel 4

Heuristische Verfahren

In diesem Kapitel beschäftigen wir uns mit dem Problem, wie man (heuristisch) zulässige Lösungen für ein gemischt ganzzahliges Programm oder kombinatorisches Optimierungsproblem finden kann. Wir werden dabei insbesondere einige Verfahren und Methoden vorstellen, die in der Praxis häufig zum Einsatz kommen. Heuristische Verfahren nützen häufig die sehr spezielle Struktur von Problemen aus, damit sie zu effizienten Verfahren werden. Wir werden hier exemplarisch auf einige grundlegende Vorgehensweise eingehen, die immer und immer wieder verwendet werden.

4.1 Der Greedy-Algorithmus

Die Idee des Greedy-Algorithmus ist eine Lösung von Null (der leeren Menge) beginnend aufzubauen und dabei im nächsten Schritt immer denjenigen Gegenstand (Variable, ...) zu nehmen, der (die) den meisten Profit verspricht (engl. greedy = gefräßig). Die Vorgehensweise des Greedy-Algorithmus läßt sich schön an Optimierungsproblemen über Unabhängigkeitssystemen erklären. Wir werden danach sehen, wie man ihn auch auf andere Probleme und Situationen anpassen kann.

Definition 4.1 *Sei E eine endliche Grundmenge. Eine Menge $\mathcal{I} \subseteq \mathcal{P}(E)$ heißt* Unabhängigkeitssystem*, falls mit $G \subseteq F \in \mathcal{I}$ auch $G \in \mathcal{I}$ folgt.*
Jede Menge $F \in \mathcal{I}$ heißt unabhängig*, alle anderen Mengen* abhängig*. Ist $F \subseteq E$, so heißt eine unabhängige Teilmenge von F* Basis von F*, falls sie in keiner anderen unabhängige Teilmenge von F enthalten ist. Sei $c : E \mapsto \mathbb{R}$ eine Gewichtsfunktion auf E. Das Problem*

$$\max_{I \in \mathcal{I}} c(I) \tag{4.1}$$

heißt Maximierungsproblem über einem Unabhängigkeitssystem.

Das Problem

$$\min_{B \text{ Basis}} c(B) \tag{4.2}$$

heißt Minimierungsproblem über einem Basissystem.

Beispiele für (4.1) sind: Das Rucksack-Problem, das Stabile-Mengen-Problem, das maximale Cliquenproblem, ...
Beispiele für (4.2) sind: Das Minimal-Aufspannende-Baum Problem, das Traveling Salesman Problem,... .
Der Greedy-Algorithmus betrachtet nun eine Sortierung der Grundelemente von E. Die Sortierung erfolgt nach einer bestimmten Gewichtung / Präferenz der Grundelemente. Diese muss nicht mit $c : E \mapsto \mathbb{R}$ übereinstimmen. Nun werden beginnend mit der leeren Menge gemäß dieser Sortierung Elemente hinzugenommen, solange die Lösung zulässig bleibt bzw. noch zu einer zulässigen Lösung ausgebaut werden kann.

Algorithmus 4.2 *Greedy-Max für Unabhängigkeitssysteme*

Input: *Ein Problem der Form (4.1).*

Output: *Zulässige Lösung $I_{Greedy} \in \mathcal{I}$.*

(1) *Sortieren: Wähle Funktion $w : E \mapsto \mathbb{R}$ und sortiere E, so dass*

$$w(e_1) \geq w(e_2) \geq \ldots \geq w(e_m),$$

wobei $m = |E|$.

(2) *Setze $I_{Greedy} = \emptyset$.*

(3) *For* $k = 1$ *To* m *Do*
Falls $w(e_k) > 0$ und $I_{Greedy} \cup \{e_k\} \in \mathcal{I}$ setze

$$I_{Greedy} = I_{Greedy} \cup \{e_k\}.$$

(4) *End For*

(5) *Gib I_{Greedy} aus.*

Der Greedy-Algorithmus für Basissysteme sieht sehr ähnlich aus:

Algorithmus 4.3 *Greedy-Min für Basissysteme*

Input: *Ein Problem der Form (4.2).*

Output: *Basis B_{Greedy} von E.*

(1) *Sortieren: Wähle Funktion $w : E \mapsto \mathbb{R}$ und sortiere E, so dass*

$$w(e_1) \leq w(e_2) \leq \ldots \leq w(e_m).$$

(2) *Setze $B_{Greedy} = \emptyset$.*

(3) *For $k = 1$ To m Do*

Falls $B_{Greedy} \cup \{e_k\}$ Basis ist oder eine Basis enthält setze

$$B_{Greedy} = B_{Greedy} \cup \{e_k\}.$$

(4) *End For*

(5) *Gib B_{Greedy} aus.*

Anwendungen

(a) Minimal aufspannende Bäume

Betrachte das Problem, in einem ungerichteten Graphen $G = (V, E)$ einen bzgl. der Gewichtung $c : E \mapsto \mathbb{R}$ minimal aufspannenden Baum zu bestimmen (d. h. eine kreisfreie Menge $B \subseteq E$ mit $|B| = |V| - 1$). Die Menge aller kreisfreien Teilmengen von E (auch Wälder genannt) ist offensichtlich ein Unabhängigkeitssystem. Wähle in Algorithmus 4.3 für die Gewichtung $w = c$. In Schritt (3) liest sich die Bedingung wie folgt: „Falls $B_{\text{Greedy}} \cup \{e_k\}$ kreisfrei“. Dann liefert Algorithmus 4.3 tatsächlich eine Optimallösung.

Satz 4.4 *Greedy-Min liefert einen minimal aufspannenden Baum in Zeit $\mathcal{O}(m \cdot n)$.*

Beweis. Wir beweisen folgende Aussage induktiv:

> Nach dem k-ten Schritt gibt es einen minimal aufspannenden Baum $\bar{B}$ mit $\bar{B} \cap \{e_1, \ldots, e_k\} = B_{\text{Greedy}} \cap \{e_1, \ldots, e_k\}$.

$k = 0$: Okay.

$k - 1 \mapsto k$: Wir unterscheiden zwei Fälle:

(a) B_{Greedy} enthält einen Kreis, d.h. $e_k \notin B_{\text{Greedy}}$.
Da $\bar{B} \cap \{e_1, \ldots, e_{k-1}\} = B_{\text{Greedy}} \cap \{e_1, \ldots, e_{k-1}\}$ enthält auch $\bar{B} \cup \{e_k\}$ einen Kreis, also $e_k \notin \bar{B}$.

(b) B_{Greedy} enthält keinen Kreis, d.h. $e_k \in B_{\text{Greedy}}$.
Gilt $e_k \in \bar{B}$ so sind wir fertig. Falls $e_k \notin \bar{B}$ so enthält $\bar{B} \cup \{e_k\}$ einen Kreis C. Da $\bar{B} \cap \{e_1, \ldots, e_{k-1}\} = B_{\text{Greedy}} \cap \{e_1, \ldots, e_{k-1}\}$ und $e_k \in B_{\text{Greedy}}$, existiert ein Index $l > k$ mit $e_l \in C$. D.h. $B' = \bar{B} \cup \{e_k\} \setminus \{e_l\}$ ist ebenfalls ein aufspannender Baum mit $c(B') \leq c(\bar{B})$ und erfüllt die Behauptung.

Zur Laufzeit: Sortieren in Schritt (1) Økostet $\mathcal{O}(m \log m)$, zum Beispiel unter Verwendung von Quicksort oder Heapsort. Schritt (3) wird maximal m-mal ausgeführt. Bleibt abzuschätzen, wie lange Schritt (3) selbst benötigt. Das Überprüfen, ob die Kantenmenge in Schritt (3) kreisfrei ist, kann mit Tiefensuche (engl. depth first search) in $\mathcal{O}(n)$ durchgeführt werden; beachte, dass B_{Greedy} maximal n Kanten enthält. Damit erhalten wir insgesamt für Algorithmus 4.3 eine Laufzeit von $\mathcal{O}(m \cdot \log m + m \cdot n) = \mathcal{O}(m \cdot n)$. □

(b) Das Rucksack-Problem

Betrachte das 0/1 Rucksack-Problem (vgl. Beispiel 1.5)

$$\begin{gathered} \max \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_j x_j \leq b \\ x_j \in \{0,1\}, \text{ für } j = 1, \ldots, n. \end{gathered}$$

Offensichtlich ist die Menge aller zulässigen Lösungen $\mathcal{F}$ ein Unabhängigkeitssystem. Wenden wir Algorithmus 4.2 auf das 0/1 Rucksack-Problem an, so kann der Algorithmus beliebig schlechte Lösungen liefern.

Beispiel 4.5

(a) Zielfunktions-Greedy
Wähle $w = c$ in Algorithmus 4.2 und betrachte folgendes Beispiel:

$$\begin{array}{lllll} \max & nx_1 + (n-1)x_2 + \ldots + (n-1)x_n + (n-1)x_{n+1} & & & \\ & nx_1 + & x_2 + \ldots + & x_n + & x_{n+1} \leq n. \end{array}$$

Algorithmus 4.2 liefert die Lösung $I_{Greedy} = \{1\}$ mit Zielfunktionswert $c_{Greedy} = n$, während die Optimallösung $c_{OPT} = n(n-1)$ ist.

(b) Gewichtsdichten-Greedy
Wähle $w_i = \frac{c_i}{a_i}$ für $i = 1, \ldots, n$ in Algorithmus 4.2 und betrachte folgendes Beispiel:

$$\begin{array}{llll} \max & x_1 & + & \alpha x_2 \\ & x_1 & + & \alpha x_2 \leq \alpha \\ & & & x_1, x_2 \in \{0,1\}. \end{array}$$

Dann liefert $c_{Greedy} = 1$, während $c_{OPT} = \alpha$ ist. Man kann jedoch zeigen, wie wir später noch sehen werden, dass der Gewichtsdichten-Greedy höchstens zweimal so schlecht ist wie die Optimallösung, wenn $x_i \in \{0,1\}$ durch $x_i \in \mathbb{Z}_+$ für alle $i = 1, \ldots, n$ ersetzt wird.

Wir sehen also, dass der Greedy-Algorithmus auch beliebig schlecht abschneiden kann. Wir wollen im Folgenden zeigen, dass der Greedy-Algorithmus dann und nur dann eine Optimallösung liefert, falls das Unabhängigkeitssystem zusätzlich folgende Eigenschaft erfüllt:

(4.3) Für alle $F \subseteq E$, B, B' Basen von F $\implies |B| = |B'|$.

In diesem Fall spricht man von einem Matroid. Um diesen Satz zeiegn zu können, benötigen wir folgende Definition:

Definition 4.6 *Für $F \subseteq E$ bezeichne*

$$r(F) := \max\left\{|B| : B \text{ ist Basis von } F\right\}$$

den Rang *von F sowie*

$$r_u(F) := \min\left\{|B| : B \text{ ist Basis von } F\right\}$$

den unteren Rang *von F.*

Bemerkung 4.7 *Für Matroide gilt offensichtlich:*

$$r_u(F) = r(F).$$

Satz 4.8 (Jenkyns, 1976 [26])

$$\min_{F \subseteq E} \frac{r_u(F)}{r(F)} \leq \frac{c(I_{Greedy})}{c(I_{OPT})} \leq 1$$

und für jedes Unabhängigkeitssystem gibt es Gewichte $c_i \in \{0,1\}$ für $i \in E$, so dass die erste Ungleichung mit Gleichheit angenommen wird.

Beweis. O.B.d.A. gelte $c_i > 0$ für $i = 1, \ldots, n$ und $c_1 \geq c_2 \geq \ldots \geq c_n$.
Wir führen ein $(n+1)$tes-Element ein mit $c_{n+1} = 0$ und setzen

$$E_i = \{1, \ldots, i\} \quad \text{sowie} \quad q = \min_{F \subseteq E} \frac{r_u(F)}{r(F)}$$

Für $F \subseteq E$ gilt:

$$c(F) = \sum_{i \in F} c_i = \sum_{i \in F} \left(\sum_{j=i}^{n} (c_j - c_{j+1}) \right) = \sum_{i=1}^{n} |F \cap E_i| \cdot (c_i - c_{i+1})$$

Da $I_{\text{OPT}} \cap E_i \subseteq I_{\text{OPT}}$ gilt $I_{\text{OPT}} \cap E_i \in \mathcal{I}$, und somit $|I_{\text{OPT}} \cap E_i| \leq r(E_i)$.
Die Vorgehensweise des Greedy-Algorithmus impliziert, dass $I_{\text{Greedy}} \cap E_i$ eine Basis von E_i ist, also $|I_{\text{Greedy}} \cap E_i| \geq r_{\text{u}}(E_i)$.
Damit gilt:

$$|I_{\text{Greedy}} \cap E_i| \geq |I_{\text{OPT}} \cap E_i| \cdot \frac{r_{\text{u}}(E_i)}{r(E_i)} \geq |I_{\text{OPT}} \cap E_i| \cdot q$$

und

$$\begin{aligned} c(I_{\text{Greedy}}) &= \sum_{i=1}^{n} |I_{\text{Greedy}} \cap E_i| \cdot (c_i - c_{i+1}) \\ &\geq \sum_{i=1}^{n} |I_{\text{OPT}} \cap E_i| \cdot q\,(c_i - c_{i+1}) \\ &= q \sum_{i=1}^{n} |I_{\text{OPT}} \cap E_i| \cdot (c_i - c_{i+1}) \\ &= q \cdot c\,(I_{\text{OPT}}) \end{aligned}$$

Die erste Ungleichung der Behauptung wäre damit bewiesen. Die zweite Ungleichung ist trivial, da jede Lösung höchstens schlechter als die Optimallösung sein kann.
Es verbleibt zu zeigen, dass das Minimum auch angenommen wird:
Sei nun $F \subseteq E$ mit $q = \frac{r_{\text{u}}(F)}{r(F)}$. O.B.d.A. sei $F = \{1, \ldots, k\}$ und $B = \{1, \ldots, p\} \subseteq F$ eine Basis von F mit $|B| = r_{\text{u}}(F)$. Setze $c_i = 1$ für $i = 1, \ldots, k$ und $c_i = 0$ sonst.
Der Greedy-Algorithmus liefert $I_{\text{Greedy}} = B$ mit $c(I_{\text{Greedy}}) = r_{\text{u}}(F) = p$, während für jede Optimallösung I_{OPT} gilt $c(I_{\text{OPT}}) = r(F)$. □

Folgerung 4.9 *Sei $\mathcal{I}$ ein Unabhängigkeitssystem auf E. Dann sind äquivalent:*

(i) $\mathcal{I}$ ist ein Matroid.

(ii) Für alle Gewichte $c \in \mathbb{R}^E$ liefert der Greedy-Algorithmus 4.2 eine optimale Lösung für $\max\{c(I) \mid I \in \mathcal{I}\}$.

(iii) Für alle „0/1“-Gewichte $c \in \{0,1\}^E$ liefert der Greedy-Algorithmus eine optimale Lösung für $\max\{c(I) \mid I \in \mathcal{I}\}$.

Beweis. Die Folgerung ergibt sich mit Bemerkung 4.7 direkt aus Satz 4.8. □

Wir haben damit durch 4.9 die Problemklasse der Matroide über einen Algorithmus 4.2 charakterisiert. Ist also kein Matroid gegeben, was zum Beispiel für das Rucksack-Problem, das Traveling Salesman Problem und für die meisten praxisrelevanten Probleme gilt, liefert der Greedy-Algorithmus keine Optimallösung, meist sogar beliebig schlechte Lösungen.

Dennoch erfreut sich der Greedy-Algorithmus in der Praxis großer Beliebtheit, denn er basiert auf einem einfach zu überschauenden Zielkriterium in Form der Funktion w und er ist lokal nicht zu entkräften.

4.2 Lokale Suche

Sobald man einmal eine zulässige Lösung gefunden hat, versucht man diese noch zu verbessern. In diesem Abschnitt behandeln wir einige Methoden und Ideen, die versuchen, dies zu leisten. Wir werden sehen, das diese Methoden auch dazu verwendet werden können, zulässige Lösungen zu finden, indem man die Unzulässigkeiten einer bereits bestehenden Lösung in die Zielfunktion mitaufnimmt und versucht diese Unzulässigkeiten zu minimieren.

Die grundlegende Vorgehensweise aller Verfahren ist gleich. Ausgehend von einer Lösung (zulässig oder unzulässig) definiert man sich eine Nachbarschaft der Lösung. In dieser Nachbarschaft sucht man nach einer besseren (oder bestmöglichen) Lösung. Hat man eine gefunden, tauscht man die alte gegen die neue Lösung und iteriert.

Sei dazu $\mathcal{F}$ die Menge von (zulässigen) Lösungen für ein Problem Π (z.B. für ein gemischt-ganzzahliges Problem, für ein Traveling Salesman Problem, ...). Für jede Lösung $S \in \mathcal{F}$ definiere eine (zulässige) Nachbarschaft $N(S) \subseteq \mathcal{F}$. Darüber hinaus sei ein Zielfunktional $w : \mathcal{F} \mapsto \mathbb{R}$ bzgl. dem wir messen, ob eine Lösung besser ist als eine andere (vgl. Gewichtung w beim Greedy-Algorithmus). Dann kann man die Grundversion der lokalen Suche wie folgt beschreiben:

Algorithmus 4.10 *Lokale Suche*

Input:

$\mathcal{F}$	*(implizit gegeben)*
$N(S)$	*für $S \in \mathcal{F}$ (implizit gegeben)*
w	*Gewichtsfunktion*
$S \in \mathcal{F}$	*(eventuell gegeben)*

Output: $S \in \mathcal{F}$.

(1) *Wähle eine Startlösung $S \in \mathcal{F}$, falls keine gegeben.*

(2) *Wähle ein $S' \in N(S)$ mit $w(S') < w(S)$.*

(3) *Falls kein solches S' existiert* – **Stop** *(Gib S aus).*

(4) *Setze $S = S'$ und gehe nach (2).*

Der Erfolg dieser Algorithmus hängt natürlich stark von der Definition der Nachbarschaft $N(\cdot)$ ab. Hier ein/zwei Beispiele.

Beispiel 4.11

- *2-Austausch beim TSP: vgl. Abbildung 4.1.*

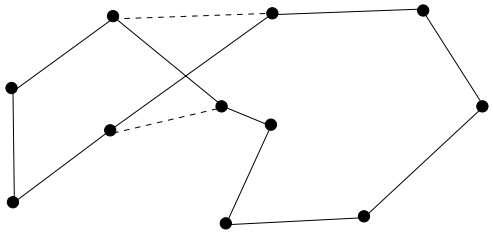

Abbildung 4.1: 2-Austausch beim TSP

- *1-Austausch / 2-Austausch beim Equipartitions-Problem: vgl. Abbildung 4.2. Als mögliches Zielfunktional kann hier beispielsweise*

$$w(S) = \sum_{e \in \delta(S)} c_e + \alpha(|S| - |V \setminus S|)^2$$

gewählt werden.

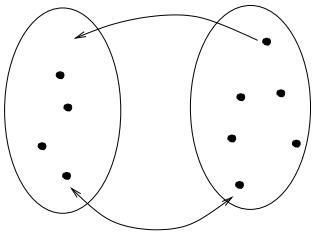

Abbildung 4.2: 1-Austausch bzw. 2-Austausch beim Equipartitions-Problem

Zielfunktional und Nachbarschaft sind natürlich stark problemabhängig. Ein Problem dieses Verfahrens ist es, dass nur Verbesserungen akzeptiert werden und damit in lokalen Minima hängen geblieben werden kann, vgl. Abbildung 4.3.

Die folgenden Verfahren versuchen dies zu vermeiden unter Beibehaltung der Grundstruktur von Algorithmus 4.10.

4.2.1 Tabu Search

Die Grundidee ist, sobald man in einem lokalen Optimum angekommen ist, die bestmögliche Lösung in der Nachbarschaft zu akzeptieren, auch wenn sie schlechter ist. Ein Problem hierbei ist das Auftreten von Kreiseln, da die Lösung der

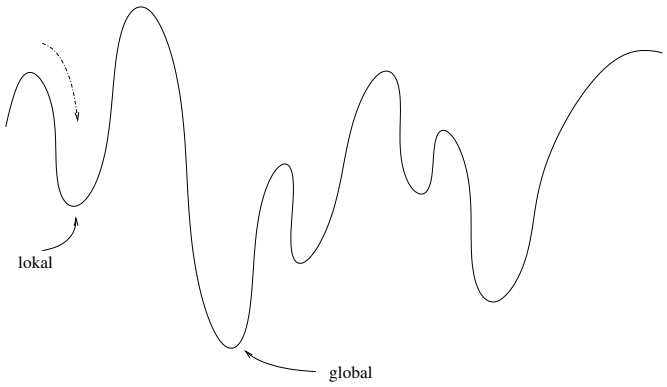

Abbildung 4.3: Lokales versus globales Minimum

Voriteration häufig in der Nachbarschaft der Folgelösung enthalten ist:

$$S^0 \to S^1 \to S^0 \to S^1 \to \cdots$$

Um ein solches Kreiseln zu vermeiden, werden gewisse Austäusche (Nachbarschaften) für tabu erklärt. Um Kreiseln vollkommen auszuschließen, müssten alle Nachbarschaften verboten werden, die zu irgendeiner vorher erzeugten Lösung führen. Dies ist allerdings sehr zeit- und speicheraufwendig. Deshalb beschränkt man sich meist auf eine Tabuliste beschränkter Größe, die die letzten Lösungen oder Austäusche beinhaltet.

Algorithmus 4.12 *Tabu Search*

Input: *wie in Algorithmus 4.10.*

Output: *wie in Algorithmus 4.10.*

(1) *Initialisiere eine leere Tabuliste T.*

(2) *Bestimme Startlösung $S \in \mathcal{F}$, falls keine gegeben.*

(3) *Solange eine Stop-Bedingung nicht erfüllt ist führe aus:*

(4) *Sei $N'(S) = N(S) \setminus T$ die Menge der Nachbarn, die nicht tabu sind.*

(5) *Bestimme $S' = \operatorname{argmin}\{w(R) : R \in N'(S)\}$.*

(6) *Setze $S = S'$ und aktualisiere die Tabuliste T.*

(7) *End Solange*

(8) *Gib die beste bis dahin gefundene Lösung S aus.*

Die Parameter, die hier zusätzlich zu denen aus Algorithmus 4.10 zu spezifizieren sind, sind:

- Tabuliste
  Je kleiner die Liste, desto größer die Gefahr des Kreiselns. Je größer die Liste, desto größer die Laufzeit.
- Stop-Bedingung
  Häufig werden hier gewählt:
  - Feste Anzahl von Iterationen.
  - Überschreitung einer gewissen Anzahl von Iterationen ohne Verbesserung.

Auch hier gilt, die einzelnen Parameter sind stark problemabhängig und es bedarf intensiven Tunings bis vernünftige Werte gefunden werden.

4.2.2 Simulated Annealing

Simulated Annealing erlaubt ebenfalls zwischendurch schlechtere Lösungen, jedoch nicht wie eben durch Steuerung einer festen Tabuliste, sondern zufallsgesteuert. Die Idee dabei ist, bessere Lösungen immer zu erlauben, schlechtere jedoch nur mit einer gewissen Wahrscheinlichkeit. Dabei ist die Wahrscheinlichkeit um so geringer je schlechter die Lösung ist. Darüber hinaus wird die Wahrscheinlichkeit, schlechtere Lösungen zu akzeptieren, im Laufe des Verfahrens immer weiter verringert, so dass der Algorithmus mit einer guten Lösung enden sollte.

Algorithmus 4.13 *Simulated Annealing*

Input: *wie in Algorithmus 4.10.*

Output: *wie in Algorithmus 4.10.*

(1) *Bestimme Startlösung $S \in \mathcal{F}$, falls keine gegeben.*

(2) *Bestimme eine Anfangstemperatur T.*

(3) *Solange Gefrierpunkt nicht erreicht führe aus:*

(4) *Führe das Folgende l-mal aus:*

(5) *Wähle einen Nachbarn S' von S zufällig.*

(6) *Setze $\Delta = w(S') - w(S)$.*

(7) *Wähle $x \in [0, 1]$ zufällig.*

(8) *Falls $\Delta < 0$ oder $x < e^{-\Delta/T}$, setze $S = S'$.*

(9) *End (l-Schleife)*

(10) *Aktualisiere T und l.*

(11) *End Solange.*

(12) *Gib die beste bis dahin gefundene Lösung S aus.*

Bemerkung 4.14

(a) *Die Idee zu diesem Verfahren kommt aus der theoretischen Physik zur Simulation von Phänomenen der statistischen Mechanik, insbesondere von Phasenübergängen wie Kristallisation von Flüssigkeiten. Daher kommen auch die Begriffe wie Temperatur und Gefrierpunkt.*

(b) *Man kann zeigen, dass bei geeigneter Wahl von l und T mit Simulated Annealing tatsächlich das Optimum gefunden werden kann, allerdings unter der Voraussetzung, das man Algorithmus 4.13 beliebig lange laufen lässt.*

(c) *Algorithmus 4.13 hat in der Regel sehr hohe Laufzeiten; es gibt keine Garantie, wann das Optimum gefunden wird. Dazu das Bild aus der Physik: Zu schnelles Abkühlen würde keinen Zustand minimaler Energie erzielen, es muss langsam abgekühlt werden, damit sich „ungewollte Strukturen (Klumpen)“ reorganisieren können.*

(d) *Viele Experimente sind notwendig, um Schritte (3) und (10) geeignet zu wählen.*

4.2.3 Genetische Algorithmen

Die grundlegende Idee genetischer Algorithmen ist, nicht nur eine Lösung zu betrachten und versuchen, diese zu verbessern, sondern eine Menge von Lösungen (Population genannt) zu betrachten und durch Kombinationen dieser Lösungen zu neuen besseren Lösungen zu gelangen. Die Grundmuster und Begriffswelt sind dabei aus der Genetik entnommen. Aus einer Population werden eine Reihe von Elternpaare (Paare von Lösungen) bestimmt, die sich kreuzen und damit ein / zwei neue Lösungen erzeugen. Diese Lösungen (evtl. zusammen mit den Eltern) werden teilweise noch mutiert (zufällig geändert). Basierend auf ihrer Fitness (Bewertungsfunktion w wie in den Abschnitten oben) wird eine neue Generation ausgewählt und der Prozess wiederholt. Im Detail sieht der Algorithmus wie folgt aus.

Algorithmus 4.15 *Genetischer Algorithmus*

Input:
$\mathcal{F}$ (implizit gegeben),
Fitnessfunktion $w : \mathcal{F} \mapsto \mathbb{R}$,
evtl. Lösungen $S_1, \ldots, S_k$ aus $\mathcal{F}$ (Population).

Output: *$S \in \mathcal{F}$.*

(1) *Bestimme Population $S_1, \ldots, S_k$, falls keine gegeben.*

(2) *Solange Stop-Kriterium nicht erfüllt führe aus:*

(3) *Elternwahl: Wähle eine Menge von Lösungspaaren aus $S_1, \ldots, S_k$.*

(4) *Kreuzung: Aus jedem Elternpaar bestimme ein oder zwei Lösungen.*

(5) *Mutation: Ändere einige der (neuen) Lösungen zufällig.*

(6) *Selektion: Bewerte die Lösungen mittels der Fitnessfunktion und wähle daraus eine neue Population $S_1, \ldots, S_k$.*

(7) *End Solange*

(8) *Gib die beste bis dahin gefundene Lösung aus.*

Bemerkung 4.16

(a) *Auch hier gilt wie bei den Verfahren vorher, die meisten Schritte sind sehr problemabhängig und können nur durch viele Tests geeignet bestimmt werden.*

(b) *Einige allgemeine Regeln zu den einzelnen Schritten sind:*

Elternwahl: Idee, wähle „fittere“ Lösungen mit höherer Wahrscheinlichkeit, z.B. S_i mit Wahrscheinlichkeit $\frac{w(S_i)}{\sum_{j=1}^{k} w(S_j)}$.

Kreuzungen: Hier werden Teillösungen, von denen man glaubt, dass sie gut sind, von den Eltern übernommen (z.B. Teilweg beim TSP) und zu neuen Lösungen verkettet.

Mutation: Meist zufällig gesteuerte Austäusche (z.B. 2-Austausch beim TSP).

Selektion: Es werden in der Regel nicht nur die (bzgl. w) besten, sondern auch mit gewissen Wahrscheinlichkeiten schlechtere Lösungen in der Population behalten.

Allen Verfahren, die wir in diesem Kapitel kennengelernt haben, ist gemein, dass sie keine Gütegarantie geben können. Sie mögen zwar oft gute Lösungen finden, aber wie gut sie tatsächlich sind oder ob sie ggf. eine Optimallösung gefunden haben, können sie nicht feststellen. Wir werden in einem späteren Kapitel noch primale Verfahren kennenlernen, die in polynomialer Zeit eine gewisse Gütegarantie geben können (sogenannte Approximationsalgorithmen). Meist sind die Garantien, die man dort erhält, für die Praxis nicht tauglich. Um wirklich (auch für die Praxis) verwertbare Garantien geben zu können, muss man sich mit der Struktur der Probleme, insbesondere mit der Struktur der Lösungsmengen der zugrundeliegenden Probleme befassen. Eine wichtige Rolle hierbei spielen Polyeder, mit denen wir uns in den nächsten Kapiteln genauer beschäftigen wollen.

Kapitel 5

Exakte Verfahren

In diesem Kapitel beschäftigen wir uns mit exakten Verfahren für diskrete Optimierungsprobleme, d.h. mit Verfahren, die den gesamten Lösungsraum absuchen und somit garantieren, eine Optimallösung zu finden, falls eine existiert. Diese Verfahren haben im schlimmsten Fall (worst case) exponentielles Laufzeitverhalten, sofern $\mathcal{P} \neq \mathcal{NP}$. Im Folgenden behandeln wir zwei klassische exakte Algorithmen, Branch-and-Bound Verfahren und Dynamische Programmierung.

5.1 Branch-and-Bound Verfahren

Der Kern von Branch-and-Bound Verfahren ist durch geschickten Einsatz von primalen und dualen Heuristiken den gesamten Suchraum so weit wie möglich einzuschränken und somit den enumerativen Teil so klein wie möglich zu halten. Die Idee ist wie folgt:
Bestimme eine zulässige Lösung mit Methoden, wie wir sie beispielsweise in Kapitel 4 kennengelernt haben. Sei z_{Best} die bis dahin gefundene beste Lösung. Falls keine primalen Verfahren vorhanden, setze $z_{Best} = +\infty$.
Gleichzeitig bestimme eine untere Schraube mit Methoden aus den vorigen Kapiteln (z. B. LP-Relaxierung, Lagrange-Relaxierung, Schnittebenenverfahren). Sei d_{Best} die beste bis dahin gefundene untere Schraube (bounding).
Gilt $d_{Best} \geq z_{Best}$, so sind wir fertig. z_{Best} ist eine (beweisbare) Optimallösung.
Andernfalls wähle eine ganzzahlige Variable $x_j \in \mathbb{Z}$ mit unterer und oberer Schranke l_j bzw. u_j, d.h. $l_j \leq x_j \leq u_j$. Generiere zwei Teilprobleme durch Wahl eines Parameters $\gamma \in \{l_j, l_{j+1}, \ldots, u_j - 1\}$. In dem ersten Teilproblem setzen wir $l_j = l_j$ und $u_j = \gamma$. In dem zweiten Problem $l_j = \gamma + 1$ und $u_j = u_j$ (branching). Das Minimum der beiden besten Lösungen aus den Teilproblemen ist offensichtlich eine Optimallösung für das Gesamtproblem.
Man hat also das Gesamtproblem ersetzt durch zwei Probleme, deren Zulässigkeitsbereich für Variable x_j jeweils eingeschränkt wurde (im Falle $u_j = l_j + 1$, z. B. falls $x_j \in \{0, 1\}$, hat man die Dimension des Problems sogar um eins reduziert.

Mit den beiden Teilproblemen fährt man entsprechend fort, bis alle Teilprobleme abgearbeitet sind. Die bis dahin beste Lösung muss eine Optimallösung sein. Im Detail sieht der Algorithmus wie folgt aus:

Algorithmus 5.1 *Branch-and-Bound Algorithmus für MIP*
Input

$$\begin{array}{ll} \min c^\top x & \\ Ax \leq b & \textit{(MIP)} \\ l \leq x \leq u & \\ x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p & \textit{mit } A \in \mathbb{Q}^{m\times n}, b, c \in \mathbb{Q}^n, l, u \in (\mathbb{Q} \cup \{\pm\infty\})^n \end{array}$$

Output
(zulässige bzw. optimale) Lösung für (MIP) oder die Meldung, dass es keine zulässige Lösung gibt.

(1) Initialisierung

$z_{IP} = +\infty$ *(Wert der besten Lösung)*

$L = \{(A, b, c, l, u)\}$ *Liste der ungelösten Probleme*

(2) Solange $L \neq \emptyset$ *führe aus*

(3) Wähle $(A, b, c, l, u) \in L$ *und setze* $L = L \setminus \{(A, b, c, l, u)\}$.

Bounding

(4) Rufe Primalverfahren für (A, b, c, l, u)*. Falls eine zulässige Lösung* $\bar{x}$ *gefunden wird und* $c^\top \bar{x} < z_{IP}$*, setze* $z_{IP} = c^\top \bar{x}$ *und* $x_{Best} = \bar{x}$*.*

(5) Rufe Dualverfahren für (A, b, c, l, u)*. Sei* d_{Best} *die beste gefundene untere Schranke.*

(6) Falls $d_{Best} \geq z_{IP}$ *gehe nach (2), es kann innerhalb der Schranken* l *und* u *keine bessere Lösung geben.*

Branching

(7) Wähle Variable $x_j \in \mathbb{Z}^{n-p}$ *und* $\gamma \in \{l_j, \ldots, u_j - 1\}$*. (*$x_j$ *und* γ *existieren, falls in (5) die LP-Relaxierung bestimmt wurde)*

Setze

$$L = L \cup \{(A, b, c, l, u'),\ (A, b, c, l', u)\}$$

mit

$$u'_i = \left\{ \begin{array}{cc} u_i & \textit{für } i \neq j \\ \gamma & \textit{für } i = j \end{array} \right\} \textit{ und } l'_i = \left\{ \begin{array}{cc} l_i & \textit{für } i \neq j \\ \gamma + 1 & \textit{für } i = j \end{array} \right\}$$

(8) End Solange

(9) Falls $z_{IP} < +\infty$, gib x_{Best} aus.

Anderenfalls die Meldung, es gibt keine zulässige Lösung.

Bemerkung 5.2

(a) Das Verfahren läuft auch ohne primale und duale Heuristiken. In diesem Fall ist es ein reines Enumerationsverfahren.

(b) Aus Lemma 2.7 wissen wir, dass es, falls $P_{I,p} \neq \emptyset$, einen ganzzahligen Punkt der Kodierungslänge höchstens $5n^4\varphi$ gibt (wobei φ die max. Kodierungslänge einer Ungleichung ist).

D.h. Algorithmus 5.1 findet in endlicher Zeit einen ganzzahligen Punkt, falls einer existiert, oder kann in endlicher Zeit entscheiden, ob es einen ganzzahligen Punkt gibt. Ist das Optimum endlich, z. B. falls $l, u \in \mathbb{Q}^n$, so wird mit Algorithmus 5.1 auch eine Optimallösung gefunden.

(c) Das Verfahren lässt sich gut in dem sog. Branch-and-Bound Baum *darstellen. Die Knoten entsprechen den Teilproblemen aus L, die Kanten geben Auskunft über die Verzweigungen.*

Man kann sich auch andere als binäre Verzweigungen vorstellen (z. B. auch Branchen auf Ungleichungen). Wichtig ist nur, dass die Vereinigung der Lösungsräume der Teilprobleme den Lösungsraum des Gesamtproblems enthält.

(d) Beachte, dass z_{IP} global *für den gesamten Baum gültig ist, d_{Best} dagegen nur* lokal *für den Teilbaum.*

(e) Wird in Schritt (5) zur Bestimmung einer unteren Schranke ein Schnittebenenverfahren verwendet, so spricht man von Branch-and-Cut Verfahren.

In diesem Fall wählt man i.d.R. in Schritt (7) eine Variable x_j, deren aktueller LP-Wert $x_j^* \notin \mathbb{Z}$ ist und setzt $\gamma = \lfloor x_j^* \rfloor$.

Branch-and-Cut Verfahren zählen derzeit zu den erfolgreichsten Methoden zur Lösung von MIPs. Nahezu jede Standardsoftware beruht auf dieser Methode.

(f) Implementierungshinweise

- *Preprocessing*
- *Auswahl der Variablen / Knoten ([30])*
- *Fixieren durch reduzierte Kosten*
- *Schnittebenen - Management*
- *Verwenden des dualen Simplex-Algorithmus*

5.2 Dynamische Programmierung

Optimallösungen für diskrete Optimierungsprobleme lassen sich häufig rekursiv aus optimalen Teillösungen zusammensetzen. Zum Beispiel muss für einen kürzesten Weg von s nach t, der über den Knoten r geht auch der Teilweg von s nach r und von r nach t optimal sein.
Dieses Prinzip der Optimalität, Teillösungen von Optimallösungen sind selbst optimal, macht sich die dynamische Programmierung zu nutzen, indem Optimallösungen sukkzessive aus Teillösungen zusammengesetzt werden. Formal wird dieses Vorgehen wie folgt eingebettet:
Wir betrachten ein dynamisches System in seinem zeitlichen Verlauf von T Perioden. In jeder Periode $t \in \{0, 1, \ldots, T\}$ kann sich das System in einer Menge von Zuständen X_t befinden. Die Zustandsvariable x_t beschreibt den Zustand des Systems zum Zeitpunkt t und fasst alle relevanten Informationen der Vergangenheit für die künftige Optimierung zusammen.
Der Zustand x_{t+1} zum Zeitpunkt $t+1$ hängt funktional vom Zustand x_t und einer Steuervariable y_t ab, d.h.

$$x_{t+1} = f_t(x_t, y_t) \quad t = 0, \ldots, T-1 \,. \tag{5.1}$$

Beachte, der Zustand x_{t+1} hängt nicht von früheren Zuständen $x_l, l < t$, ab. Die Steuervariablen seien Elemente einer Menge S_t.
Ebenso gibt es eine zu minimierende Zielfunktion,

$$g_t(x_t, y_t) \quad t = 0, \ldots, T \tag{5.2}$$

die additiv über die Zeit ist. D.h. der Gesamtwert G ergibt sich aus

$$G = \sum_{t=0}^{T-1} g_t(x_t, y_t) + g_T(x_T) \,, \tag{5.3}$$

wobei $g_T(x_T)$ Terminalkosten am Ende des Prozesses bezeichnen.

Definition 5.3 *(Dynamisches Programm)*
Das folgende Problem heißt Dynamisches Programm

$$\begin{aligned}
&\min \sum_{t=0}^{T-1} g_t(x_t, y_t) + g_T(x_T) \\
&x_{t+1} = f_t(x_t, y_t) \\
&x_0 \in X_0 \text{ (meist vorgegeben = Anfangszustand)} \\
&x_t \in X_t \qquad t = 1, \ldots, T \\
&y_t \in S_t \qquad t = 0, \ldots, T-1
\end{aligned}$$

Eine Entscheidungsfolge $\Pi = (y_0, \ldots, y_{T-1})$ *heißt* Strategie.
Eine Strategie heißt zulässig, *falls* $y_t \in S_t$ *und* x_t, *definiert durch (5.1), aus* X_t *ist.*

Ein Dynamisches Programm ist also nichts anderes als ein Optimierungsproblem über einem dynamischen System.
Das Prinzip der Optimalität lässt sich an einem Dynamischen Programm sehr einfach formulieren.

Prinzip der Optimalität für Dynamische Programme:
Sei $\Pi^* = (y_0^*, \ldots, y_{T-1}^*)$ eine optimale Strategie für 5.3. Betrachte nun das Teilproblem, in dem wir am Zeitpunkt t beginnend (d.h. x_t ist der Anfangszustand) die Kosten für die verbleibenden Perioden $l = t+1, \ldots, T$ minimieren möchten. Dann ist die Teilstrategie $(y_t^*, \ldots, y_{T-1}^*)$ optimal für dieses Teilproblem.

In anderen Worten, die Entscheidung y_t zum Zeitpunkt t hängt nicht von den vorhergehenden Entscheidungen $y_1, \ldots, y_{t-1}$ ab. Dieses Prinzip kann man sich nun algorithmisch zunutze machen, in dem man von hinten beginnend ($t = T$) sich optimale Teilstrategien konstruiert und diese zu optimalen Strategien für frühere Perioden zusammensetzt:

Algorithmus 5.4 *DP Algorithmus*
Input Ein dynamisches Programm (DP) wie in Definition 5.3
Output Optimalwert von (DP) und eine optimale Strategie $\Pi^* = (y_0, \ldots, y_{T-1})$

(1) Setze $J_T(x_T) = g_T(x_T)$ *für* $x_T \in X_T$

(2) *For* $t = T-1$ *to* 0 *do*

$$J_t(x_t) = \min_{y_t \in S_t} \Big(g_t(x_t, y_t) + J_{t+1}(\overbrace{f_t(x_t, y_t)}^{x_{t+1}}) \Big)$$

(3) Gib $J_0(x_0)$ *und* $\Pi^* = (y_0^*, \ldots, y_{T-1}^*)$ *aus, wobei* y_t^* *jeweils die rechten Seiten in (2) minimieren.*

Satz 5.5 *Algorithmus 5.4 löst (DP).*

Beweis. Seien $J_t^*(x_t) = \min_{\Pi}(g_T(x_T) + \sum_{k=t}^{T-1} g_k(x_k, y_k))$ die Werte der optimalen Teilstrategien. Wir zeigen per Induktion nach t, dass $J_t(x_t) = J_t^*(x_t)$ für alle $x_t \in X_t$.
Der Fall $t = T$ ist klar.
Nehmen wir also an, dass für ein $t < T$ und alle $x_{t+1} \in X_{t+1}$ gilt $J_{t+1}^*(x_{t+1}) = J_{t+1}(x_{t+1})$.
Dann gilt mit $\Pi^t = (y_t, \Pi^{t+1})$ für alle x_t.

$$
\begin{aligned}
J_t^*(x_t) &= \min_{(y_t,\Pi^{t+1})} (g_t(x_t,y_t) + g_T(x_T) + \sum_{k=t+1}^{T-1} g_k(x_k,y_k)) \\
\text{(Prinzip der Optimalität)} &= \min_{y_t\in S_t} g_t(x_t,y_t) + \min_{\Pi^{t+1}} (g_T(x_T) + \sum_{k=t+1}^{T-1} g_k(x_k,y_k)) \\
&= \min_{y_t\in S_t} g_t(x_t,y_t) + J_{t+1}^*(x_{t+1}) \\
\text{(Ind. ann.)} &= \min_{y_t\in S_t} g_t(x_t,y_t) + J_{t+1}(x_{t+1}) \\
&= J_t(x_t)\,.
\end{aligned}
$$

□

Anwendungen

Beispiel 5.6 *Kürzeste Wege*
Betrachte einen gerichteten Graphen $D = (V, A)$ mit nichtnegativen Gewichten $c_{ij} \geq 0$. Gesucht sind für einen Knoten s die kürzesten Wege von s zu allen anderen Knoten $i \in V$.
Zur Berechnung dieser Wege betrachte die Funktion

$$D_k(i) = \begin{array}{l}\textit{Länge eines kürzesten Weges von } s \textit{ nach } i,\\ \textit{der über maximal } k \textit{ Knoten läuft.}\end{array}$$

Dann gilt die Rekursion

$$D_k(j) = \min\left\{D_{k-1}(j), \min_{i\in V(j)} (D_{k-1}(i) + c_{ij})\right\} \tag{5.4}$$

Berechnen wir diese Werte für k von 1 bis $n-1$ für alle $j \in V$, so enthält am Ende $D_{n-1}(j)$ den Wert eines kürzesten Weges von s nach j. Die Laufzeit des Algorithmus beträgt $O(m \cdot n)$.
Dieser Algorithmus ist ein DP Algorithmus im Sinne von Algorithmus 5.4.
Betrachte dazu das folgende Dynamische Programm

$$
\begin{aligned}
T &= n \\
X_t &= \{P_j \mid P_j \textit{ ist ein kürz. Weg von } s \textit{ nach } j \textit{ über max. } t \textit{ Knoten}\} \\
&\quad \cup\ \{\emptyset\} \\
S_t &= V \\
f_t(P_j,i) &= \begin{cases} P_i & \textit{, falls } ij \in A, \\ P_j & \textit{, falls } i = j, \\ \emptyset & \textit{, sonst.} \end{cases} \\
g_t(P_j,i) &= \begin{cases} c_{ij} & \textit{, falls } ij \in A, \\ 0 & \textit{, falls } i = j, \\ \infty & \textit{, sonst.} \end{cases}
\end{aligned}
$$

Damit liest sich Algorithmus 5.4 wie folgt

$$J_0(P_i) = \begin{cases} c_{si}, & \textit{falls } si \in A, \\ +\infty, & \textit{sonst.} \end{cases} \tag{5.5}$$
$$J_0(\emptyset) = +\infty$$

(Setze $P_i = \{si\}$*, falls* $J_0(P_i) = c_{si}, P_i = \emptyset$*, sonst.)*

$$\begin{aligned} J_t(P_j) &= \min_{i \in V} \left(g_t(P_j, i) + J_{t-1}(f_t(P_j, i))\right) \\ &= \min\left(\min_{i \in V(\delta(j))} c_{ij} + J_{t-1}(P_i), 0 + J_{t-1}(P_j), \infty\right). \end{aligned} \tag{5.6}$$

Setze

$$P_j = \begin{cases} P_i \cup \{ij\}, & \textit{falls } J_t(P_i) = c_{ij} + J_{t-1}(P_i), \\ P_j, & \textit{falls } J_t(P_j) = J_{t-1}(P_j). \end{cases}$$

was genau (5.4) entspricht. Beachte auch, dass $P_i \cup \{ij\} = P_j$ *wieder einen Weg ergibt, da* $c_{ij} \geq 0$*.*

Beispiel 5.7 *Das* $0/1$ *Rucksackproblem*
Betrachte folgendes $0/1$ *Rucksackproblem*

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} c_i y_i \\ & \sum_{i=1}^{n} a_i y_i \leq b \\ & y_i \in \{0,1\} \quad i = 1, \ldots, n \end{aligned}$$

Das $0/1$ *Rucksackproblem kann auch als Dynamisches Programm aufgefasst werden. Sei dazu*

$$\begin{aligned} T &= n+1 \textit{ (für jede Variable eine Periode plus Slackvariable)} \\ x_j &= \textit{Restkapazität in Periode } j \textit{ (Zustandsvariable)} \\ y_j &= \textit{Steuervariablen} \end{aligned}$$

Übergangsfunktion f *aus (5.1) ist*

$$x_{j+1} = f_j(x_j, y_j) = x_j - a_j y_j$$

Für die Zielfunktion (5.2) gilt

$$g_j(x_j, y_j) = c_j y_j$$

Darüber hinaus haben wir

$$\begin{aligned} X_j &= \{0, 1, \ldots, b\}, \quad j = 1, \ldots, n+1 \\ X_1 &= \{b\}, \end{aligned}$$

sowie

$$\begin{aligned} S_j &= \{0,1\}, \quad \textit{falls } x_j \geq a_j \\ S_j &= \{0\}, \quad \textit{sonst.} \end{aligned}$$

Beachte, dass S_j vom Zustand x_{j-1} der Vorperiode abhängt, was aber der Gültigkeit der entwickelnden Theorie und Verfahren keinen Abbruch tut.
Das 0/1 Rucksackproblem liest sich nun als (DP) wie folgt

$$\max \sum_{j=1}^{T} g_j(x_j, y_j) = \sum_{j=1}^{n} c_j y_j$$

$$\begin{aligned} x_{j+1} &= x_j - a_j y_j \quad j = 1, \ldots, n+1 \\ x_1 &= b \\ x_j &\in X_j \\ y_j &\in S_j \end{aligned}$$

Algorithmus 5.4 liest sich entsprechend

$$J_{n+1}(d) = 0 \quad \textit{für alle } d = 0, 1, \ldots, b \tag{5.7}$$

$$\begin{aligned} J_i(d) &= \max_{y_i \in S_i} (g_i(d, y_j) + J_{i+1}(f_i(d, y_i))) \\ &= \begin{cases} J_{i+1}(d), & \textit{falls } d < a_i, \\ \max\{J_{i+1}(d), c_i + J_{i+1}(d - a_i)\}, & \textit{falls } d \geq a_i, \end{cases} \end{aligned} \tag{5.8}$$

für $i = n, n-1, \ldots, 1$. Die Optimallösung ist dann $\max_d J_1(d)$.

Beachte Algorithmus 5.4 hat Laufzeit $O(n \cdot b)$, ist also pseudo-polynomial und für kleine rechte Seiten durchaus verwendbar.

5.3 Primale Methoden

In diesem Kapitel beschäftigen wir uns mit exakten primalen Verfahren zur Lösung von ganzzahligen Programmen. Genauer gesagt, behandeln wir sogenannte Augmentierungsverfahren, das heißt Verfahren, die ausgehend von einer zulässigen Lösung durch iteratives Verbessern eine optimale Lösung zu finden versuchen. Wir haben uns in in Kapitel 4 mit Heuristiken beschäfigen, die ebenfalls als lokale Verbesserungsverfahren interpretiert werden können, aber im Allgemeinen keine Optimallösungen garantieren können. In diesem Kapitel wollen wir uns genauer mit der Menge der zugrundeliegenden Verbesserungsvektoren, auch Augmentierungsvektoren genannt, beschäftigen. Dies wird uns auf den Begriff der Testmenge führen. Wir werden sehen, dass Testmengen in engem Zusammenhang zu Hilbert-Basen stehen, wie wir sie in Kapitel 2.4 eingeführt haben.

Wir werden eine vergleichbare Äquivalenz zur Separierung und Optimierung, vgl. die Vorlesung „Einführung in die Optimierung“ beweisen, in dem wir zeigen, dass das Finden einer Optimallösung polynomial äquivalent zum Augmentierungsproblem ist, das heißt zu dem Problem, gegeben eine zulässige Lösung, entscheide, ob die Lösung optimal ist, wenn nicht gib einen Verbesserungsvektor an. Diese theoretischen Untersuchungen führen uns auch auf neue algorithmische primale Ansätze. Wir werden in diesem Zusammenhang die „Integral Basis Method“, die auf Arbeiten von Weismantel zurückgeht, genauer untersuchen.

Kapitel 6

Approximationsalgorithmen

Allen primalen Verfahren, die wir bislang in Kapitel 4 kennengelernt haben, ist gemein, dass sie keine Gütegarantie geben können. Sie mögen zwar oft gute Lösungen finden, aber wie gut sie tatsächlich sind oder ob sie ggf. eine Optimallösung gefunden haben, können sie nicht beweisen. In diesem Kapitel betrachten wir exemplarisch noch einige Verfahren, die in polynomialer Zeit immer eine gewisse Lösungsgarantie erreichen.

Definition 6.1 *(Approximations-Algorithmen) Sei* Π *ein diskretes Optimierungsproblem und A ein Algorithmus zur Lösung von* Π*. Bezeichne* $c_{opt}(I)$ *den Optimalwert für* $I \in \Pi$ *und* $c_A(I)$ *den Wert, den Algorithmus A liefert.*

(a) A heißt ϵ-Approximations-Algorithmus*, falls A polynomiale Laufzeit hat und* $c_A(I)$ *höchstens* ϵ*-mal schlechter als* $c_{opt}(I)$ *ist für alle* $I \in \Pi$*. D.h. für*

$$\textit{Minimierungsprobleme: } c_A \leq \epsilon\, c_{opt} \textit{ mit } \epsilon \geq 1$$
$$\textit{Maximierungsprobleme: } c_A \geq \epsilon\, c_{opt} \textit{ mit } \epsilon \leq 1$$

ϵ heißt die Gütegarantie *von A.*

(b) A heißt polynomiales Approximationsschema (PAS)*, falls für jedes feste* $\epsilon > 0$ *A in polynomialer Laufzeit (polynomial in* $\langle I \rangle$ *für* $I \in \Pi$*) eine Gütegarantie von* $1 + \epsilon$ *(bzw.* $1 - \epsilon$*) hat.*

(c) A heißt vollpolynomiales Approximationsschema (FPAS)*, falls die Laufzeit polynomial in* $\langle I \rangle$ *und* $\frac{1}{\epsilon}$ *ist.*

Ein FPAS ist das Beste, was man erreichen kann für ein $\mathcal{NP}-$schweres Problem wie der folgende Satz zeigt.

Satz 6.2 *Sei* Π *ein diskretes Optimierungsproblem. Gibt es für* Π *ein FPAS, das auch polynomial in* $\langle \epsilon \rangle$ *ist, so gibt es einen polynomialen Algorithmus für* Π*.*

Beweis. Wir zeigen die Aussage für Minimierungsprobleme (für Maximierungsprobleme geht es analog).
Sei A ein FPAS für Π, das auch polynomial in $\langle\epsilon\rangle$ ist. O.B.d.A. sei die Zielfunktion c ganzzahlig. Wir konstruieren einen polynomialen Algorithmus wie folgt:

1. Setze $\epsilon = \frac{1}{2}$ und wende A auf $I \in \Pi$ an. Wir erhalten einen Wert $c_{A_\epsilon}(I)$.
2. Setze $\delta = \frac{1}{c_{A_\epsilon}(I)+1}$.
3. Wende A auf I mit Approximationsgüte δ an. Wir erhalten einen Lösungswert $c_{A_\delta}(I)$.

Obiger Algorithmus zur Berechnung von $c_{A_\delta}(I)$ ist polynomial. Wir zeigen nun noch, dass $c_{opt}(I) = c_{A_\delta}(I)$.

$$\begin{aligned} c_{opt}(I) &\leq c_{A_\delta}(I) \leq (1+\delta)c_{opt}(I) = (1+\tfrac{1}{c_{A_\epsilon}(I)+1})c_{opt}(I) \\ &< (1+\tfrac{1}{c_{opt}(I)})c_{opt}(I) = c_{opt}(I)+1. \end{aligned}$$

Im vorletzten Schritt wurde $c_{opt}(I) < c_{A_\epsilon}(I) + 1$ benutzt. Da c ganzzahlig ist, folgt die Behauptung. □

Im Folgenden wollen wir uns noch Approximations-Algorithmen bzw. FPAS für verschiedene diskrete Optimierungsprobleme anschauen. Wir werden dabei einige interessante Algorithmenideen kennenlernen, die sich auf andere Probleme übertragen lassen. Wir beginnen mit einem Beispiel aus einem großen Anwendungsfeld für Approximations-Algorithmen, den Scheduling-Problemen.

6.1 Randomisierte Rundeverfahren

Die Idee vieler Approximations-Algorithmen ist das Potential der LP-Relaxierungen zu nutzen und diese geschickt zu runden. Der Erfolg dieser Verfahren hängt von dem Geschick ab, die gerundete Lösung geeignet abschätzen zu können. Hier helfen häufig randomisierte Techniken. Wir erläutern die Ideen an einem Beispiel aus dem Scheduling-Bereich, einem der klassischen Bereiche für Approximationsalgorithmen.

Beispiel 6.3 *Betrachte m identische Maschinen $M_1, \ldots, M_m, m \geq 2$, und n Jobs $J_1, \ldots, J_n$. Jeder Job J_i hat Strafkosten e_i bei Ablehnung und Produktionszeiten p_{ij} auf Maschine M_j. Jeder Job kann beliebig unterbrochen und weitergeführt werden. Für jeden Job soll entschieden werden, ob er angenommen wird oder nicht und die angenommenen Jobs sollen so auf die Maschinen verteilt werden, dass der Makespan (Produktionszeit der langsamsten Maschine) minimiert wird.*

Formulierung als MIP:

$$\begin{aligned} x_{ij} &= \textit{Anteil von Job } j \textit{ auf Maschine } i \\ y_j &= \begin{cases} 1, & \textit{falls Job } i \textit{ angenommen wird} \\ 0, & \textit{sonst.} \end{cases} \\ T &= \textit{Makespan} \end{aligned}$$

$$\begin{aligned} \min \quad & T + \sum_{j=1}^{n} (1 - y_j) e_j \\ & \sum_{j=1}^{n} p_{ij} x_{ij} \leq T \qquad i = 1, \ldots, m \\ & \sum_{i=1}^{m} p_{ij} x_{ij} \leq T \qquad j = 1, \ldots, n \\ & \sum_{i=1}^{m} x_{ij} = y_j \qquad j = 1, \ldots, n \\ & x_{ij} \geq 0 \qquad i = 1, \ldots, m,\ j = 1, \ldots, n \\ & y_j \in \{0, 1\} \quad j = 1, \ldots, n. \end{aligned}$$

Beachte, dass obiges Modell das Scheduling-Problem tatsächlich korrekt beschreibt, da die Jobs beliebig unterbrechbar sind. Man kann zeigen, dass obiges Schedulingproblem $\mathcal{NP}$-schwer ist.

Satz 6.4 *Das Scheduling-Problem aus Beispiel 6.3 ist $\mathcal{NP}$-schwer.*

Beweis. Zum Beweis siehe [25]. □

Betrachte folgenden Algorithmus zur Lösung des Scheduling-Problems:

Algorithmus 6.5 Deterministischer Rundealgorithmus

(1) *Bestimme Lösung x^*, y^* der LP-Relaxierung des Scheduling-Problems.*

(2) *Wähle α mit $0 < \alpha < 1$.*

(3) *Für jeden Job j führe aus:*

(4) *Ist $y_j^* \leq \alpha$ setze $y_j = x_{ij} = 0$ für alle i.*

(5) *Ist $y_j^* > \alpha$ setze $y_j = 1$ und $x_{ij} = \frac{x_{ij}^*}{y_j^*}$ für alle i.*

(6) *End Für*

(7) *Gib y, x aus.*

Satz 6.6 *Algorithmus 6.5 ist ein 2-Approximations-Algorithmus.*

Beweis. Es gilt (vgl. Schritt (4) und (5))

$$(1 - y_j) \leq \frac{1}{1-\alpha}\,(1 - y_j^*)$$

und

$$x_{ij} \leq \frac{1}{\alpha}\, x_{ij}^*$$

und damit

$$\begin{aligned} T + \sum_{j=1}^{n}(1 - y_j)e_j &\leq \frac{1}{\alpha}\, T^* + \frac{1}{1-\alpha}\sum_{j=1}^{n}(1 - y_j^*)e_j \\ &\leq \max\{\frac{1}{\alpha}, \frac{1}{1-\alpha}\}(T^* + \sum_{j=1}^{n}(1 - y_j^*)e_j) \\ &\leq \max\{\frac{1}{\alpha}, \frac{1}{1-\alpha}\}c_{opt}. \end{aligned}$$

Für $\alpha = \frac{1}{2}$ erhält man die Behauptung. □

Durch Randomisieren kann man die Gütegarantie sogar noch auf $\frac{e}{e-1} \approx 1.58$ verbessern:

Algorithmus 6.7 Randomisierter Rundealgorithmus

(1) *Wie in 6.5.*

(2) *Für jedes $\alpha \in [\frac{1}{e}, 1] \cap \{y_1^*, \ldots, y_n^*\}$ führe Schritte (3)-(5) in 6.5 aus.*

(3) *Gib die beste der n gefundenen Lösungen aus.*

Satz 6.8 *Algorithmus 6.7 ist* 1.58 *approximativ.*

Beweis. Betrachte zunächst eine gleichverteilte Zufallsvariable α auf dem Intervall $[\frac{1}{e}, 1]$. Dann gilt

$$E(T) = \frac{e}{e-1}\int_{\frac{1}{e}}^{1} T d\alpha \leq \frac{e}{e-1}\int_{\frac{1}{e}}^{1} \frac{1}{\alpha} T^* d\alpha = \frac{e}{e-1} T^*.$$

Ferner haben wir

$$\begin{aligned} E(\sum_{j=1}^{n}(1-y_j)e_j) &= \sum_{j=1}^{n} e_j \text{ Prob}(y_j^* \leq \alpha) \\ &= \sum_{j=1}^{n} e_j \int_{\max\{\frac{1}{e}, y_j^*\}}^{1} \frac{e}{e-1} d\alpha \\ &\leq \sum_{j=1}^{n} e_j \int_{y_j^*}^{1} \frac{e}{e-1} d\alpha \\ &= \sum_{j=1}^{n} \frac{e}{e-1} \cdot e_j(1-y_j^*) \\ &= \frac{e}{e-1} \sum_{j=1}^{n} e_j(1-y_j^*). \end{aligned}$$

D.h.

$$E(T + \sum_{j=1}^{n} e_j(1-y_j)) \leq \frac{e}{e-1} c_{opt}$$

und damit haben wir einen randomisierten $\frac{e}{e-1}$ Approximations-Algorithmus.
Zur Derandomisierung beachte, dass verschiedene Lösungen nur dann auftreten, falls $\alpha \in \{y_1^*, \ldots, y_n^*\}$ und damit genügt es für $\alpha \in [\frac{1}{e}, 1] \cap \{y_1^*, \ldots, y_n^*\}$ gerundete Lösungen zu berechnen. □

6.2 Ein FPAS für das Rucksackproblem

In diesem Abschnitt wollen wir uns ein FPAS für das Rucksackproblem betrachten. Die Idee des Algorithmus ist es, sich auf die Gegenstände zu konzentrieren, die den maximalen Gewinn versprechen und dieses Teilproblem annähernd optimal zu lösen. Bei geeigneter Definition von "groß" und "annähernd" kann dieses Problem mittels Dynamischer Programmierung optimal gelöst werden. Der Rest wird dann per Greedy-Algorithmus aufgefüllt:

Algorithmus 6.9 *(FPAS für das* $0/1$ *Rucksackproblem)*

Input: $a_j, c_j \in \mathbb{Z}_+, j = 1, \ldots, n$. *O.B.d.A* $\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \ldots \geq \frac{c_n}{a_n}$, $b \in \mathbb{Z}_+$ *und zwei Parameter* s, t.

Output: *Approx. Lösung des* $0/1$ *Rucksackproblems*

(geeignete Wahl von s *und* t *liefern Gütegarantie* ϵ*, siehe Satz 6.11).*

(1) *Abschätzung der Optimallösung.*

Sei k maximaler Index mit $\sum_{j=1}^{k} a_j \leq b$.
Setze $c_{est} = \sum_{j=1}^{k+1} c_j$.

(2) *Zerlegung der Indexmenge*

$L := \{j \in \{1, \ldots, n\} \mid c_j \geq t\}$*(large indices),*
$S := \{j \in \{1, \ldots, n\} \mid c_j < t\}$*(small indices).*

(3) *Löse folgende spezielle Rucksack-Probleme für $d = 0, 1, \ldots, \lfloor \frac{c_{est}}{s} \rfloor$:*

$$(GKP_d) \qquad \begin{aligned} \min & \sum_{j \in L} a_j x_j \\ & \sum_{j \in L} \lfloor \frac{c_j}{s} \rfloor x_j = d \\ & x_j \in \{0,1\}, \qquad j \in L \end{aligned}$$

(4) *Für $d = 0, 1, \ldots, \lfloor \frac{c_{est}}{s} \rfloor$ führe aus*

(5) *Sei $x_j^d, j \in L$ die Optimallösung von (GKP_d)*

(6) *Falls $\sum_{j \in L} a_j x_j^d \leq b$ gilt, wende den Gewichtsdichtengreedy an auf*

$$(KP_d) \qquad \begin{aligned} \max & \sum_{j \in S} c_j x_j \\ & \sum_{j \in S} a_j x_j \leq b - \sum_{j \in L} a_j x_j^d \ (=: b_d) \\ & x_j \in \{0,1\}, \qquad j \in S \end{aligned}$$

Sei $x_j^d, j \in S$, die Greedy-Lösung.
Dann ist $x_j^d, j = 1, \ldots, n$ eine Lösung des $0/1$ Rucksackproblems.

(7) *End For (d)*

(8) *Gib die beste der maximal $(\lfloor \frac{c_{est}}{s} \rfloor + 1)$ gefundenen Lösungen aus.*

Satz 6.10 *Sei c_{IK} der durch Algorithmus 6.9 gefundene Lösungswert. Dann gilt*

$$c_{IK} \geq c_{opt} - (\frac{s}{t} c_{opt} + t).$$

Beweis. Sei $x_1^*, \ldots, x_n^*$ eine Optimallösung des 0/1 Rucksackproblems. Setze $d = \sum_{j \in L} \lfloor \frac{c_j}{s} \rfloor x_j^*$. Dann gilt

$$d \leq \lfloor \frac{1}{s} \sum_{j \in L} c_j x_j^* \rfloor \leq \lfloor \frac{1}{s} c_{opt} \rfloor \leq \lfloor \frac{1}{s} c_{est} \rfloor,$$

wobei zu beachten ist, dass c_{est} größer oder gleich als der LP-Wert ist. Also ist in Algorithmus 6.9 eine Kandidatenlösung $\bar{x}_1, \ldots, \bar{x}_n$ mit

$$\sum_{j \in L} \lfloor \frac{c_j}{s} \rfloor \bar{x}_j = d$$

betrachtet worden, da in diesem Fall die Lösungsmenge nicht leer ist.
Dann gilt

$$\text{(A)} \qquad c_{IK} \geq \sum_{j=1}^{n} c_j \bar{x}_j = c_{opt} - (\sum_{j \in L} c_j x_j^* - \sum_{j \in L} c_j \bar{x}_j) - (\sum_{j \in S} c_j x_j^* - \sum_{j \in S} c_j \bar{x}_j).$$

Wir erhalten

$$\begin{aligned} \text{(B)} \quad \sum_{j \in L} c_j x_j^* - \sum_{j \in L} c_j \bar{x}_j &\leq s \underbrace{\sum_{j \in L} \lfloor \frac{c_j}{s} \rfloor x_j^*}_{=d} + \sum_{j \in L} (c_j - s \lfloor \frac{c_j}{s} \rfloor) x_j^* - s \underbrace{\sum_{j \in L} \lfloor \frac{c_j}{s} \rfloor \bar{x}_j}_{=d} \\ &= \sum_{j \in L} \underbrace{(c_j - s \lfloor \frac{c_j}{s} \rfloor)}_{\leq s} x_j^* \leq s \sum_{j \in L} x_j^* \leq \frac{s}{t} \sum_{j \in L} c_j x_j^* \leq \frac{s}{t}\, c_{opt}. \end{aligned}$$

Um den zweiten Term in (A) abzuschätzen, benötigen wir folgende Beziehung

$$(*) \qquad c_{Greedy} \geq c_{opt} - \max\{c_j \mid j = 1, \ldots, n\}.$$

Gilt $\sum_{j=1}^{n} a_j \leq b$, so ist dies offensichtlich richtig. Andernfalls gilt

$$0 \leq b - \sum_{j=1}^{k} a_j < a_{k+1}$$

und damit

$$c_{opt} \leq \sum_{j=1}^{k} c_j + c_{k+1} \frac{b - \sum_{j=1}^{n} a_j}{a_{k+1}} < c_{Greedy} + c_{k+1}.$$

Mit $(*)$ folgt

$$\underbrace{c_{Greedy}}_{(KP_d)} = \sum_{j \in S} c_j \bar{x}_j > \underbrace{c_{opt}}_{(KP_d)} - \max\{c_j \mid j \in S\} \geq \underbrace{c_{opt}}_{(KP_d)} - t.$$

Es folgt daraus

(C) $$t > \underbrace{c_{opt}}_{(KP_d)} - \underbrace{c_{Greedy}}_{(KP_d)} \geq \sum_{j \in S} c_j x_j^* - \sum_{j \in S} c_j \bar{x}_j.$$

(A), (B) und (C) zusammen ergeben

$$c_{IK} \geq c_{opt} - (\frac{s}{t} c_{opt} + t).$$

□

Satz 6.11 *Sei $\epsilon > 0$. Setze $s = (\frac{\epsilon}{3})^2 c_{est}$ und $t = \frac{\epsilon}{3} c_{est}$. Dann gilt*

(a) Die Laufzeit von Algorithmus 6.9 ist $O(n \log n) + O(n(\frac{1}{\epsilon})^2)$.

(b) Für den Wert der Optimallösung gilt

$$c_{IK} \geq (1 - \epsilon) c_{opt}.$$

Also ist Algorithmus 6.9 ein FPAS für das 0/1 Rucksackproblem.

Beweis.

Ad (a): Schritt (1) und (6) (einmalig) benötigen zur Sortierung $O(n \log n)$. Berechnung von s und t ist $O(\log \epsilon + n)$. Lösung von Schritt (3) benötigt $O(n \lfloor \frac{c_{est}}{s} \rfloor) = O(\frac{n}{\epsilon^2})$, siehe Kapitel 5.2. In Schritt (4) wird $O(\frac{1}{\epsilon^2})$ mal der Greedy-Algorithmus mit Laufzeit $O(n)$ angewendet, macht $O(\frac{n}{\epsilon^2})$. Insgesamt beläuft sich die Laufzeit also auf $O(n \log n + \frac{n}{\epsilon^2})$.

Ad (b): Zunächst gilt $\sum_{j=1}^{k} c_j \leq c_{opt}$ und $c_{k+1} \leq c_{opt}$ und damit $c_{est} \leq 2 c_{opt}$ sowie $t \leq \frac{2\epsilon}{3} c_{opt}$. Mit Satz 6.11 erhalten wir

$$c_{IK} \geq c_{opt} - (\frac{s}{t} c_{opt} + t) \geq c_{opt} - (\frac{\epsilon}{3} c_{opt} + \frac{2\epsilon}{3} c_{opt}) = (1 - \epsilon) c_{opt}.$$

□

6.3 Primal-Dual Verfahren

Die Idee dieser Verfahren ist, ausgehend von einer zulässigen Lösung des Dualen der LP-Relaxierung eines Ganzzahligen Programms eine zulässige primale Lösung zu generieren. Wir wissen aus der Vorlesung „Einführung in die Optimierung“, ein Paar von Lösungen x und y ist optimal für das primale bzw. duale lineare Programm genau dann, wenn die Bedingungen des Satzes vom schwachen komplementären Schlupfes erfüllt sind. Wir wissen, dass für lineare Programme,

welche bekanntlich polynomial lösbar sind, diese Bedingungen auch tatsächlich erfüllt werden können. Für ganzzahlige Programme ist dies nicht zu erwarten, sofern $\mathcal{P} \neq \mathcal{NP}$. Die Idee ist nun die komplementären Schlupfbedingungen zu relaxieren und sich auf nur eine der beiden Bedingungen zu konzentrieren. Wir werden das Verfahren an einer sehr allgemeinen Klasse von Problemen, dem sog. Hitting Set Problem, erklären. Wir werden gleich sehen, dass dieses Problem viele bekannte polynomiale als auch $\mathcal{NP}$-schwere Probleme beinhaltet.

Definition 6.12 Hitting Set Problem
Gegeben ist eine Grundmenge E und eine Liste von Teilmengen $T_1, \ldots, T_p \subseteq E$ sowie nicht-negative Gewichte c_e, $e \in E$. Finde eine Teilmenge $A \subseteq E$ minimalen Gewichtes, so dass $A \cap T_i \neq \emptyset$ für $i = 1, \ldots, p$ (d.h. A „hits“ (trifft) jedes T_i).

Das Hitting Set Problem beinhaltet unter anderem folgende Probleme

- Kürzeste-Wege-Problem in ungerichteten Graphen
- Knotenüberdeckungs-Problem
- Minimal-Aufspannende-Baum Problem
- Minimales Arboreszenz-Problem
- Steinerbaum-Problem
- Set Covering Problem (ist sogar äquivalent dazu)
- Survivable Network Design Problem
- Perfektes Matching Problem

Das Hitting Set Problem kann als ganzzahliges Programm wie folgt modelliert werden:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & \\ & \sum_{e \in T_i} x_e \geq 1 & \text{für } i = 1, \ldots, p \\ & x_e \in \{0, 1\} & \text{für } e \in E. \end{aligned} \tag{6.1}$$

Die LP-Relaxierung von (6.1) lautet

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & \\ & \sum_{e \in T_i} x_e \geq 1 & \text{für } i = 1, \ldots, p \\ & x_e \geq 0 & \text{für } e \in E, \end{aligned} \tag{6.2}$$

und das duale lineare Programm zu (6.2) ist

$$\begin{aligned} \min \quad & \sum_{i=1}^{p} y_i \\ & \sum_{i:\, e \in T_i} y_i \leq c_e \quad \text{für } e \in E, \\ & y_i \geq 0 \qquad\qquad \text{für } i = 1, \ldots, n. \end{aligned}$$

Sei x der Inzidenzvektor einer Menge $A \subseteq E$ und y eine dual zulässige Lösung, so lauten die komplementären Schlupf-Bedingungen:

$$e \in A \implies \sum_{i:\, e \in T_i} y_i = c_e \tag{6.3}$$

$$y_i > 0 \implies |A \cap T_i| = 1 \tag{6.4}$$

(6.3) werden auch die primalen und (6.4) die dualen komplementäre Schlupf-Bedingungen genannt.

Die Idee ist nun, sich auf die primalen Bedingungen zu konzentrieren und die dualen zu relaxieren.
Betrachte eine zulässige duale Lösung y (diese ist einfach zu finden, z. B. $y = 0$, da $c \geq 0$) und setze

$$A := \{e \in E \mid \sum_{i:\, e \in T_i} y_i = c_e\}.$$

Ist A zulässig, so haben wir ein A gefunden, das (6.3) erfüllt. Ist A unzulässig, so kann es keine zulässige Lösung geben, die (6.3) erfüllt. In diesem Fall kann jedoch y erhöht werden.
Da A unzulässig ist, existiert ein T_k mit $A \cap T_k = \emptyset$. Wir nennen T_k verletzt. Erhöhen wir y_k, so verbessert sich die duale Zielfunktion. Um duale Zulässigkeit beizubehalten, kann y_k maximal auf

$$y_k = \min_{e \in T_k} \left\{ c_e - \sum_{i \neq k:\, e \in T_i} y_i \right\} \tag{6.5}$$

erhöht werden. Beachte $y_k > 0$, da $A \cap T_k = \emptyset$ und somit $c_e - \sum_{i \neq k:\, e \in T_i} y_i \geq c_e - \sum_{i:\, e \in T_i} y_i > 0$ für alle $e \in T_k$.
Für diesen Wert y_k gibt es mindestens ein $e \in E \setminus A$, nämlich argmin in (6.5), das zu A hinzugefügt werden kann. Dieses Verfahren wird nun wiederholt bis A zulässig ist.

Algorithmus 6.13 Primal-Dual Algorithmus (Grundversion)

(1) *Setze* $y = 0$.

(2) *Setze* $A = \emptyset$

(3) *Solange* $\exists k : A \cap T_k = \emptyset$ *führe aus*

(4) *Erhöhe* y_k *bis* $\exists e \in T_k : \sum_{i:\, e \in T_i} y_i = c_e$.

(5) *Setze* $A = A \cup \{e\}$.

(6) *End Solange*

(7) *Gib A (und y) aus.*

Um die Laufzeit von Algorithmus 6.13 abzuschätzen, beachte, dass die Schritte (3) bis (6) höchstens $|E|$-mal durchlaufen werden, da $A \subseteq E$, und dass Schritt (4) maximal $O(|E|)$ Zeit benötigt. Die Gesamtlaufzeit hängt also von Schritt (3) ab. Nehmen wir an, wir haben ein Orakel, das uns, gegeben eine Menge $A \subseteq E$, entscheidet, ob A zulässig ist, und falls nicht, ein T_k mit $A \cap T_k = \emptyset$ zurückgibt, so haben wir einen orakel-polynomialen Algorithmus.

Im Folgenden nehmen wir an, dass dieses Orakel uns immer eine (bzgl. Inklusion) minimale verletzte Menge zurückgibt. Im Falle von Netzwerkentwurfsproblemen, in denen die $T_i = \delta(S_i)$ für ein $S_i \subset V$ sind, beziehen wir die Minimalität auf die zugehörige Knotenmenge S_i. Wir nennen diese Regel die Minimale Verletztheitsregel. Wir werden sehen, dass diese Mengen in den nun folgenden Anwendungen einfach zu bestimmen sind und wir somit einen polynomialen Algorithmus 6.13 erhalten.

Um die Güte der Lösung A in Schritt (7) von Algorithmus 6.13 abschätzen zu können, beobachten wir, dass aufgrund von (6.3) gilt:

$$c(A) = \sum_{e \in A} c_e = \sum_{e \in A} \sum_{i:\, e \in T_i} y_i = \sum_{i=1}^{p} |A \cap T_i|\, y_i. \tag{6.6}$$

Gelingt es uns nun ein α zu finden, so dass

$$|A \cap T_i| \leq \alpha \text{ für alle } i = 1, \ldots, p \text{ mit } y_i > 0, \tag{6.7}$$

so ist Algorithmus 6.13 ein α-Approximationsalgorithmus.

Ein triviales α ist

$$\alpha = \max_{i=1,\ldots,p} |T_i|. \tag{6.8}$$

Diese Beobachtung liefert uns zum Beispiel für das Knotenüberdeckungs-Problem einen 2-Approximationsalgorithmus, da $|T_i| = 2$ für alle $i = 1, \ldots, p$.

Satz 6.14 *Algorithmus 6.13 ist ein 2-Approximationsalgorithmus für das Knotenüberdeckungs-Problem.*

Im Allgemeinen ist jedoch (6.8) keine Konstante und (6.7) auch nicht. Betrachte zum Beispiel Algorithmus 6.13 für das Kürzeste-Wege-Problem auf einem Stern, vgl. Abbildung 6.1.

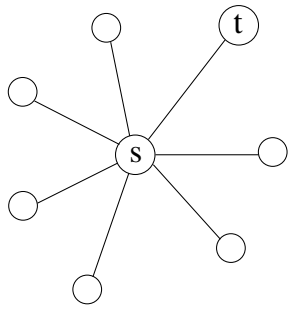

Abbildung 6.1: Beispiel für nicht konstantes α

Hier ist es möglich, dass A am Ende von Algorithmus 6.13 alle Kanten beinhaltet, jedoch die Kante st genügen würde.

Wir verfeinern daher unseren Algorithmus und eliminieren am Ende alle unnötigen Kanten:

Algorithmus 6.15 Verbesserter Primal-Dual Algorithmus

(1) *Setze* $y = 0$.

(2) *Setze* $A = \emptyset$

(3) *Setze* $l = 0$

(4) *Solange* $\exists k : A \cap T_k = \emptyset$ *führe aus*

(5) *Setze* $l = l + 1$.

(6) *Erhöhe* y_k *bis* $\exists e_l \in T_k : \sum_{i:\, e \in T_i} y_i = c_{e_l}$.

(7) *Setze* $A = A \cup \{e_l\}$.

(8) *End Solange*

(9) *For* $j = l$ *Downto 1 Do*

(10) *Falls* $A \setminus \{e_j\}$ *zulässig, setze* $A = A \setminus \{e_j\}$.

(11) *End For*

(12) *Gib A (und y) aus.*

Schritte (9) bis (11) nennen wir Rückwärtslöschung.
Durch diese Erweiterung erhalten wir eine neue interessante Abschätzung für $|A \cap T_i|$ für alle $i = 1, \ldots, p$.

Definition 6.16 *Sei $A \subseteq E$ unzulässig. Dann heißt $B \subseteq E$* minimale Erweiterung von *A, falls gilt:*

(a) B ist zulässig.

(b) $A \subseteq B$.

(c) Für alle $e \in B \setminus A$ gilt: $B \setminus \{e\}$ ist unzulässig.

In Abbildung 6.2 ist eine minimale Augmentierung für das Kürzeste-Wege-Problem angedeutet, die gestrichelten Kanten sind die Kanten in A, B enthält zusätzliche alle durchgezogenen Kanten.

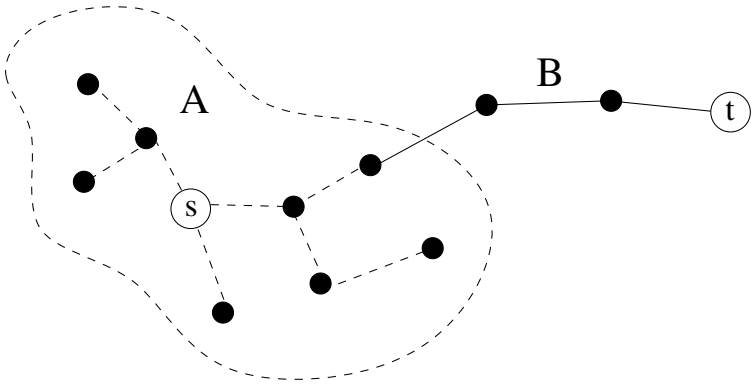

Abbildung 6.2: Beispiel einer minimalen Augmentierung

Betrachte nun noch einmal Algorithmus 6.15 und sei A_f die Lösung, die am Schluss in Schritt (12) ausgegeben wird. Betrachte in Schritt (10) die Situation, dass e_j nicht gelöscht werden kann. Dann ist $A = \{e_1, \ldots, e_{j-1}\}$ unzulässig und $B = A_f \cup \{e_1, \ldots, e_{j-1}\}$ eine minimale Augmentierung von $\{e_1, \ldots, e_{j-1}\}$. Ferner gilt:

$$|A_f \cap T_i| \leq |B \cap T_i| \quad \text{für alle } i = 1, \ldots, p.$$

Letzteres gilt sicher auch, wenn wir das Maximum über alle möglichen minimalen Augmentierungen nehmen. Sei

$$\beta = \max_{\substack{A \subset E \\ \text{unzulässig}}} \max_{\substack{B\text{:min.} \\ \text{Augm. von } A}} |B \cap T(A)|, \tag{6.9}$$

wobei $T(A)$ die von Algorithmus 6.15 in Schritt (4) gewählte Menge T_k bezeichne, wenn dem Orakel als Input die Menge A gegeben wird.

Damit erhalten wir

Satz 6.17 *Algorithmus 6.15 ist ein β-Approximationsalgorithmus für das Hitting Set Problem.*

Folgerung 6.18 *Für das Kürzeste-Wege-Problem gilt $\beta = 1$. D.h. Algorithmus 6.15 löst das Kürzeste-Wege-Problem in polynomialer Zeit.*

Beweis. Sei A eine beliebige unzulässige Menge. D.h. A zerfällt in mindestens 2 Komponenten, eine davon, sagen wir T_s, enthält den Startknoten s, eine andere den Endknoten t, die wir mit T_t bezeichnen. Aufgrund unserer minimalen Verletztheitsregel wird in Schritt (4) T_s vom Orakel zurückgegeben. Offensichtlich gilt für jede minimale Augmentierung B von A, dass $|B \cap \delta(T_s)| = 1$ gilt. Daraus folgt $\beta = 1$. □

Folgerung 6.19 *Für das Minimalkosten-Arboreszenz-Problem gilt $\beta = 1$. D.h. Algorithmus 6.15 löst das Minimalkosten-Arboreszenz-Problem in polynomialer Zeit.*

Beweis. Zur Wiederholung, das Minimalkosten-Arboreszenz-Problem ist das Problem, gegeben ein gerichteter Graph $G = (V, E)$ mit nicht-negativen Bogengewichten und ein bestimmter Knoten $r \in V$, Wurzel genannt, finde einen aufspannenden Baum mit minimalen Kosten, so dass es einen gerichteten Weg von r zu allen anderen Knoten gibt.

Sei nun A eine beliebige Teilmenge von E. Das Verletztheitsorakel mit der minimalen Verletztheitsregel in Schritt (4) von Algorithmus 6.15 kann so implementiert werden, dass zunächst alle stark zusammenhängenden Komponenten bestimmt werden und dann überprüft wird, ob es eine Komponente S gibt mit $r \notin S$ und $\delta^-(S) \cap A = \emptyset$. Gibt es kein solches S, so enthält A offensichtlich eine r-Arboreszenz.
Andernfalls erhalten wir in Schritt (4) eine stark zusammenhängende Komponente S mit $r \notin S$ und $\delta^-(S) \cap A = \emptyset$. Offensichtlich enthält jede minimale Augmentierung B von A genau einen Bogen aus $\delta^-(S)$, da dadurch alle Knoten in S erreichbar sind. Damit folgt $\beta = 1$. □

Wir wollen uns zum Abschluss dieses Kapitels noch eine weitere Verfeinerung von Algorithmus 6.13 bzw. Algorithmus 6.15 ansehen. Für Graphenprobleme, die mehrere Ziel- und/oder Quellknoten haben (wie zum Beispiel für das Matching-Problem oder das Steinerbaum-Problem) reichen die bisherigen Argumente nicht aus, da eine minimale Augmentierung immer eine nicht-konstante Anzahl an Kanten in den jeweiligen Schnitten enthalten kann.

Die Idee ist nun, sich nicht nur auf eine verletzte Menge in Schritt (4) zu konzentrieren, sondern auf alle gleichzeitig und gleichzeitig alle zugehörigen dualen Variablen zu erhöhen. Die konstante Approximation erhält man, wie wir später sehen werden, schließlich durch Durchschnittsbildung. Zunächst der Algorithmus.

Algorithmus 6.20 Simultaner Primal-Dual Algorithmus

(1) *Setze* $y = 0$.

(2) *Setze* $A = \emptyset$

(3) *Setze* $l = 0$

(4) *Solange* A *nicht zulässig führe aus*

(5) *Setze* $l = l + 1$.

(6) $\mathcal{V}(A) =$ *Menge aller minimalen verletzten Mengen* S *von* A.

(7) *Erhöhe* y_S *gleichmäßig für alle* $S \in \mathcal{V}(\mathcal{A})$ *bis* $\exists e_l \notin A : \sum_{i:\, e \in T_i} y_i = c_{e_l}$.

(8) *Setze* $A = A \cup \{e_l\}$.

(9) *End Solange*

(10) *For* $j = l$ *Downto* 1 *Do*

(11) *Falls* $A \setminus \{e_j\}$ *zulässig, setze* $A = A \setminus \{e_j\}$.

(12) *End For*

(13) *Gib* A *(und* y*) aus.*

Zur Abschätzung der Güte von Algorithmus 6.20 bezeichne in Iteration j mit ϵ_j die Erhöhung aller dualen Variablen y_i mit $T_i \in \mathcal{V}_j$, wobei $\mathcal{V}_j$ die Menge der minimal verletzten Mengen in Iteration j sei. Dann gilt

$$y_i = \sum_{j:\, T_i \in \mathcal{V}_j} \epsilon_j,$$

und damit (vgl. Schritt (7))

$$\sum_{i=1}^{p} y_i = \sum_{j=1}^{l} |\mathcal{V}_j|\, \epsilon_j.$$

Für die Kosten von A_f ergibt sich damit, vgl. (6.6):

$$\begin{aligned} c(A_f) &= \sum_{i=1}^{p} |A_f \cap T_i| y_i \\ &= \sum_{i=1}^{p} |A_f \cap T_i| \sum_{j:\, T_i \in \mathcal{V}_j} \epsilon_j \\ &= \sum_{j=1}^{l} \Big(\sum_{T_i \in \mathcal{V}_j} |A_f \cap T_i| \Big) \epsilon_j. \end{aligned}$$

Das heißt wir erhalten einen γ-Approximationsalgorithmus, falls

$$\sum_{T_i \in \mathcal{V}_j} |A_f \cap T_i| \leq \gamma |\mathcal{V}_j|. \tag{6.10}$$

Wenden wir nun die gleiche Argumentation wie in (6.9) und Satz 6.17 an, so erhalten wir folgenden Satz.

Satz 6.21 *Algorithmus 6.20 ist ein γ-Approximationsalgorithmus, falls*

$$\sum_{T_i \in \mathcal{V}(A)} |B \cap T_i| \leq \gamma |\mathcal{V}(A)| \tag{6.11}$$

für alle unzulässigen Mengen A und alle minimalen Augmentierungen B von A.

Wir wollen nun diesen Algorithmus verwenden, um Approximationsresultate (genauer gesagt 2-Approximationen) für Netzwerkentwurfsprobleme zu zeigen, die folgende Struktur haben:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ & \sum_{e \in \delta(S)} x_e \geq f(S) \quad \text{für alle } S \subseteq V, \\ & x_e \in \{0,1\} \qquad\quad \text{für } e \in E. \end{aligned} \tag{6.12}$$

$f : 2^V \mapsto \{0,1\}$ sei dabei eine 0/1 Funktion mit folgenden Eigenschaften:

(a) $f(V) = 0$.

(b) f ist maximal, das heißt

$$\forall A, B \subseteq V, A \cap B = \emptyset \text{ gilt} : f(A \cup B) \leq \max\{f(A), f(B)\}.$$

(c) f ist symmetrisch, das heißt: $f(S) = f(V \setminus S)$.

Wir nennen eine Funktion mit diesen Eigenschaften echt.

Beachte, dass die Mengen T_i in Algorithmus 6.20 assoziiert werden können mit Knotenmengen $S_i \subseteq V$, wobei $T_i = \delta(S_i)$. Unser Ziel (6.11) mit $\gamma = 2$ zu zeigen, liest sich damit wie folgt:

$$\sum_{S \in \mathcal{V}(A)} |\delta_B(S)| \leq 2|\mathcal{V}(A)|. \tag{6.13}$$

Bevor wir (6.13) für Probleme der Form (6.12) zeigen, geben wir zunächst zwei Beispiele:

Beispiel 6.22 Das Steinerbaum-Problem
*Gegeben sei ein ungerichteter Graph $G = (V, E)$ und eine Teilmenge der Knoten $T \subseteq V$ (*Terminalmenge *genannt). Finde eine Kantenmenge A, die T aufspannt, das heißt eine Kantenmenge A, so dass alle Paare $s, t \in T$ durch einen Weg in (V, A) verbunden sind.*
Setze $f(S) = 1$, falls $S \cap T \neq \emptyset$ und $(V \setminus S) \cap T \neq \emptyset$, sowie $f(S) = 0$ andernfalls. Dann ist f echt und (6.12) ist eine korrekte Modellierung des Steinerbaum-Problems (siehe Übung).

Beispiel 6.23 Das T-join Problem
Sei $T \subseteq V$ eines Graphen $G = (V, E)$ und $|T|$ gerade. Finde eine Kantenmenge $A \subseteq E$ minimalen Gewichtes (minimal bzgl. einer Funktion $c : E \mapsto \mathbb{R}_+$), so dass jeder Knoten in T ungeraden Grad und jeder Knoten nicht in T geraden Grad hat.
Setze $f(S) = 1$, falls $|S \cap T|$ ungerade, und $f(S) = 0$, falls $|S \cap T|$ gerade. Dann ist f echt und (6.12) ist eine korrekte Modellierung des T-join Problems (siehe Übung).

Um (6.13) zeigen zu können benötigen wir zwei Lemmata.

Lemma 6.24 *Sei f echt und $A \subseteq E$ beliebig. Dann gilt:*

(a) A ist zulässig genau dann, wenn für alle zusammenhängenden Komponenten C von (V, A) gilt $f(C) = 0$.

(b) Die minimalen verletzten Mengen von A sind die zusammenhängenden Komponenten C von (V, A) mit $f(C) = 1$.

Beweis. (a) ist klar, vgl. (6.12).
Um (b) zu zeigen, betrachte eine verletzte Menge S mit $f(S) = 1$ und $\delta_A(S) = \emptyset$. Offensichtlich besteht S aus der Vereinigung von zusammenhängenden Komponenten von S. Da f maximal ist, muss mindestens für eine dieser Komponenten C gelten $f(C) = 1$. Diese ist damit ebenfalls verletzt. Das heißt nur zusammenhängende Komponenten können minimal verletzte Mengen sein und diese sind dies natürlich auch. □

Lemma 6.25 *Sei $f : 2^V \mapsto \{0,1\}$ symmetrisch. Dann gilt: f ist maximal genau dann, wenn f* komplementär *ist, das heißt:*

$$\forall S \subseteq V \text{ und } A \subseteq S \text{ mit } f(A) = f(S) = 0 \text{ folgt: } f(S \setminus A) = 0.$$

Beweis.

„⇒“ Sei $S \subseteq V$, $A \subseteq S$ und $f(S) = f(A) = 0$. Zu zeigen ist $f(S \setminus A) = 0$.

Da f symmetrisch, folgt $f(V \setminus S) = 0$. Da f maximal und $f(A) = 0$ folgt damit $f(A \cup (V \setminus S)) = 0$. Wieder mit dem Argument, dass f symmetrisch ist, erhalten wir somit $f(V \setminus (A \cup (V \setminus S)) = 0$. Die letztere Menge ist jedoch gerade $S \setminus A$, woraus die Behauptung folgt.

„⇐“ Seien $A, B \subseteq V$ mit $A \cap B = \emptyset$ beliebig. Zu zeigen ist $f(A \cup B) \leq \max\{f(A), f(B)\}$.

Gilt $f(A) = 1$ oder $f(B) = 1$ so ist nichts zu zeigen. Sei also $f(A) = f(B) = 0$. Zu zeigen ist $f(A \cup B) = 0$. Da f symmetrisch, folgt $f(V \setminus A) = 0$. Wenden wir nun die Voraussetzung auf $S = V \setminus A$ und $A = B$ an, so gilt $f((V \setminus A) \setminus B) = 0$. Letztere Menge ist gleich $V \setminus (A \cup B)$, woraus mit der Symmetrie von f die Behauptung folgt.

□

Nun sind wir soweit, um unseren Hauptsatz zu zeigen.

Satz 6.26 *Algorithmus 6.20 ist ein 2-Approximationsalgorithmus für das ganzzahlige Programm (6.12), falls $f : 2^V \mapsto \{0,1\}$ echt ist.*

Beweis. Betrachte eine beliebige unzulässige Menge A. Sei $\mathcal{V}(A)$ die Menge aller minimal verletzten Mengen, siehe Schritt (6) von Algorithmus 6.20. Aus Lemma 6.24 (b) wissen wir, dass $f(S) = 1$ für alle $S \in \mathcal{V}(A)$. Betrachte weiter eine beliebige minimale Augmentierung B von A und den Graphen (V, B).

Wir bestimmen aus (V, B) einen neuen Graphen, indem wir jede zusammenhängende Komponente in (V, A) zu einem Knoten kontrahieren. Sei (H, F) der so entstehende Graph. Da B (kanten-)minimal ist, ist (H, F) ein Wald und es besteht eine 1-1 Beziehung zwischen den Kanten $B \setminus A$ und F. Ferner gehört jeder Knoten $v \in H$ zu einer zusammenhängenden Komponente S_v von (V, A). Sei $d_F(v)$ der Grad des Knoten v in (H, F), d.h. $d_F(v) = |\delta_B(S_v)|$.

Sei weiter W die Menge aller Knoten, deren zugehörige Zusammenhangskomponente verletzt ist, das heißt $\mathcal{V}(A) = \{S_w \mid w \in W\}$. Um (6.13) zu zeigen, genügt es also

$$\sum_{w \in W} d_F(v) \leq 2|W| \tag{6.14}$$

zu zeigen.

Für den Beweis von 6.14 zeigen wir zunächst, dass für alle Blätter v von (H, F), das heißt für alle Knoten $v \in H$ mit $d_F(v) = 1$, gilt $f(S_v) = 1$.

Angenommen dem wäre nicht so. Sei v ein solcher Knoten und e die zu v inzidente Kante, vgl. Abbildung 6.3, und sei ferner C die Zusammenhangskomponente in (V, B), die S_v enthält (beachte, S_v ist Zusammenhangskomponente in (V, A)).

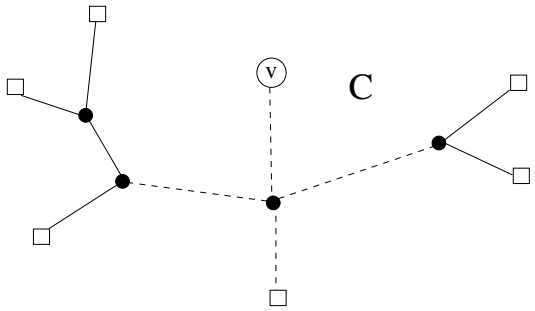

Abbildung 6.3: Illustration des Grad-Arguments im Beweis von Satz 6.20

Da B zulässig, gilt $f(C) = 0$. Da nach Annahme $f(S_v) = 0$, folgt mit Lemma 6.25, dass $f(C \setminus S_v) = 0$ gilt. Da jedoch B minimal ist, ist $B \setminus \{e\}$ unzulässig, was bedeutet, dass $f(S_v) = 1$ oder $f(C \setminus S_v) = 1$ gelten muss (vgl. Lemma 6.24), ein Widerspruch.

Also wissen wir, jedes Blatt in (H, F) gehört zu W. Nun gilt:

$$\begin{aligned}\sum_{w \in W} d_F(v) &= \sum_{w \in H} d_F(v) - \sum_{w \notin W} d_F(v) \\ &\leq (2|H| - 2) - \sum_{w \notin W} d_F(v) \\ &\leq (2|H| - 2) - 2(|H| - |W|) \\ &= 2|W| - 2,\end{aligned}$$

was zu zeigen war. □

Folgerung 6.27 *Algorithmus 6.20 ist ein 2-Approximationsalgorithmus für das Steinerbaum-Problem.*

Folgerung 6.28 *Algorithmus 6.20 ist ein 2-Approximationsalgorithmus für das T-join Problem.*

Man kann sich leicht überlegen, dass die relevanten Schritte (6) und (7) in Algorithmus 6.20 tatsächlich in polynomialer Zeit (in der Tat in linearer Zeit) für die angegebenen Probleme implementiert werden können.

Derselbe Algorithmus oder zumindest dieselbe Idee kann für weitere kombinatorische Optimierungsprobleme angewendet werden. Dazu zählen Verallgemeinerungen des Steinerbaum-Problems, das Perfekte Matching Problem, das Aufspannende Baum Problem und viele mehr. Mehr Informationen hierzu sind in [16] zu finden.
Es gibt viele weitere Approximationsresultate mit vielen interessanten Ideen, wie schon erwähnt sind ein großes Feld Scheduling-Probleme. Weitere bekannte Beispiele beinhalten das euklidischer TSP, das MAX-CUT Problem oder Standortplanungsprobleme um nur einige Beispiele zu nennen. Weiterführende Literatur hierzu ist zum Beispiel das Buch von Dorit Hochbaum mit dem Titel „Approximation Algorithms“.

Literaturverzeichnis

[1] E. Balas. Facets of the knapsack polytope. *Mathematical Programming*, 8:146 – 164, 1975.

[2] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed $0-1$ programs. *Mathematical Programming*, 58:295–324, 1993.

[3] J.F. Benders. Partitioning procedures for solving mixed variables programming. *Numerische Mathematik*, 4:238–252, 1962.

[4] C. Berge. Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind (Zusammenfassung). In *Wissenschaftliche Zeitschrift*, Mathematisch-Naturwissenschaftliche Reihe. Martin Luther Universität Halle-Wittenberg, 1961.

[5] R. Borndörfer. *Aspects of Set Packing, Partitioning, and Covering*. PhD thesis, Technische Universität Berlin, 1998.

[6] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305 – 337, 1973.

[7] V. Chvátal. On certain polytopes associated with graphs. *Journal on Combinatorial Theory B*, 18:305 – 337, 1975.

[8] S.A. Cook. The complexity of theorem-proving procedures. In *Theory Computing*, ACM, Proc. 3rd ann. ACM Sympos., pages 151 – 158. Shaker Heights, Ohio, 1971.

[9] G.B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[10] J. Edmonds and R. Giles. A min-max relation for submodular functions on graphs. In P.L. Hammer, editor, *Studies in Integer Programming*, volume 1, pages 185 – 204. North-Holland, Amsterdam, 1977.

[11] C.E. Ferreira. *On Combinatorial Optimization Problems Arising in Computer System Design*. PhD thesis, Technische Universität Berlin, 1994.

[12] Jr. L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[13] D. Fulkerson, A.J. Hoffman, and R. Oppenheim. On balanced matrices. *Mathematical Programming Study*, 1:120–132, 1974.

[14] D.R. Fulkerson. Blocking and Anti-Blocking Pairs of Polyhedra. *Mathematical Programming*, 1:168–194, 1971.

[15] D.R. Fulkerson. Packing rooted directed cuts in a weighted directed graph. *Mathematical Programming*, 6:1 – 13, 1974.

[16] M.X. Goemans and D.P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D. Hochbaum, editor, *Approximation Algorithms*, 1997.

[17] J.L. Goffin and J.P. Vial. Convex nondifferentiable optimization: A survey focussed on the analytic center cutting plane method. Technical Report 99.02, Logilab, Universite de Geneve, 1999.

[18] R.E. Gomory. An algorithm for the mixed integer problem. Technical Report RM-2597, The RAND Cooperation, 1960.

[19] R.E. Gomory. Solving linear programming problems in integers. In R. Bellman and M. Hall, editors, *Combinatorial analysis, Proceedings of Symposia in Applied Mathematics*, volume 10, Providence RI, 1960.

[20] R.E. Gomory. An algorithm for integer solutions to linear programming. In R.L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269 – 302, New York, 1969. McGraw-Hill.

[21] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988.

[22] P.L. Hammer, E.L. Johnson, and U.N. Peled. Facets of regular 0-1 polytopes. *Mathematical Programming*, 8:179 – 206, 1975.

[23] J.B. Hiriart-Urruty and C. Lemarechal. Convex analysis and minimization algorithms, part 2: Advanced theory and bundle methods. In *Grundlehren der mathematischen Wissenschaften*, volume 306. Springer-Verlag, 1993.

[24] A.J. Hoffmann and J.B. Kruskal. Integral boundary points of convex polyhedra. In H. W. Kuhn and A. W. Tucker, editors, *Linear inequalities and related systems*, pages 223–246. Princeton University Press, Princeton, NJ, 1956.

[25] S. Hoogeveen, M. Skutella, and G. Woeginger, 2001. Personal Communication with M. Skutella.

[26] T.A. Jenkins. The efficacy of the "greedy" algorithm. *Proc. 7th southeast. Conf. Comb., Graph Theory, Comput.*, pages 341 – 350, 1976.

[27] D. Klabjan, G.L. Nemhauser, and C. Tovey. The complexity of cover inequality separation. *Operations Research Letters*, 23:35 – 40, 1998.

[28] H. Marchand, A. Martin, R. Weismantel, and L.A. Wolsey. Cutting planes in integer and mixed integer programming. Technical Report CORE DP9953, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, 1999.

[29] H. Marchand and L.A. Wolsey. The $0-1$ knapsack problem with a single continuous variable. *Mathematical Programming*, 85:15 – 33, 1999.

[30] A. Martin. Integer programs with block structure. Habilitations-Schrift, Technische Universität Berlin, 1998.

[31] M.W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215, 1973.

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

[33] M.W. Padberg. $(1, k)$-configurations and facets for packing problems. *Mathematical Programming*, 18:94–99, 1980.

[34] B. Polyak. A general method of solving extremum problems (in russian). *Doklady Akademii Nauk SSR*, 174:33 – 36, 1967. [English translation in USSR Computational Mathematics and Mechanical Physics 9 (1969), 14 – 29].

[35] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, Chichester, 1986.

[36] R. Weismantel. On the $0/1$ knapsack polytope. *Mathematical Programming*, 77:49–68, 1997.

[37] L.A. Wolsey. Faces of linear inequalities in 0-1 variables. *Mathematical Programming*, 8:165 – 178, 1975.