Diskrete Optimierung

Skriptum einer 2-stündigen Vorlesung
Sommersemester 2003

Prof. Dr. Peter Hauck

Diskrete Mathematik
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
<hauck@informatik.uni-tuebingen.de>
http://www-dm.informatik.uni-tuebingen.de

LaTeX-Version des Skriptums erstellt von Matthias Müller

Inhaltsverzeichnis

Inhaltsverzeichnis **i**

1 Einleitung **1**

2 Lineare Optimierung **3**

2.1 Standardprogramme 3

2.2 Dualität 5

2.3 Alternativsätze 7

2.4 Der Hauptsatz der linearen Optimierung 11

2.5 Zulässige und optimale Lösungen 14

2.6 Simplex-Algorithmus 21

3 Ganzzahlige lineare Optimierung **31**

3.1 Branch-and-Bound 34

3.2 ILP und LP-Relaxation; Terminierung von Branch-and-Bound . 37

3.3 Schnittebenen-Verfahren (Cutting Planes Algorithm) 42

3.4 Der Lenstra-Algorithmus 46

4 Optimierungsprobleme auf Graphen **47**

4.1 Graphentheoretische Grundlagen 47

4.2 Minimale aufspannende Bäume (Minimum Spanning Trees) . . . 49

4.3 Kürzeste Wege 52

4.4 Das Traveling-Salesman-Problem (TSP) 56

5 Das Knapsack-Problem **59**

6 Weitere Themen **63**

Literaturverzeichnis **65**

Kapitel 1

Einleitung

Optimierungsprobleme stellen eines der wichtigsten Anwendungsgebiete von Methoden der diskreten Mathematik und Informatik dar. Zur Einstimmung geben wir einige Beispiele.

Beispiel 1.1:
Ablaufplanung: Gegeben sei eine Anzahl von Arbeitsaufträgen (Jobs), die alle auf einer Maschine bearbeitet werden. Jeder Job belegt die Maschine für einen Tag. $f(i)$ sei dabei der Fertigstellungstermin für den Job i, bei dessen Überschreitung eine Strafe $s(i)$ zu bezahlen ist.
Ziel: Bestimmung der Bearbeitungsreihenfolge der Jobs, sodass die Gesamtstrafe minimal wird.

Beispiel 1.2:
Transportproblem: Gegeben seien mehrere Lager $L_1, \ldots, L_m$, sowie mehrere Verkaufsstellen $V_1, \ldots, V_n$. Dabei benötigt die Verkaufsstelle V_j insgesamt b_j Mengeneinheiten einer Ware, von der jeweils a_i Mengeneinheiten im Lager L_i vorhanden sind. Die Kosten für den Transport der Ware von Lager L_i zur Verkaufsstelle V_j sei c_{ij}.
Ziel: Bestimmung eines Transportplans, sodass die Transportkosten minimal werden.

Beispiel 1.3:
Netzwerkproblem: Gegeben Sei ein Netzwerk von Leitungen, die über Knoten miteinander verbunden sind. Dabei besitzt eine Leitung zwischen zwei Knoten eine gewisse Kapazität.
Ziel: Optimierung des Transports des Transportgutes (z.B. Flüssigkeiten, Strom, Information) zwischen zwei Knoten.

Beispiel 1.4:
Knapsack-Problem: Gegeben sei ein Transportflugzeug mit maximaler Ladung K. Dieses soll die (unteilbaren) Waren $a_1, \ldots, a_n$ transportieren, von denen jede Ware a_i einen bestimmten Wert c_i besitzt.
Ziel: Auswahl der Waren, sodass deren Gewicht K nicht übersteigt und der Gesamtwarenwert maximal wird.

BEISPIEL 1.5:
Traveling Salesman Problem: Ein Handlungsreisender will eine Rundreise durch n Städte durchführen, jede Stadt dabei nur einmal besuchen und am Ende schließlich zum Ausgangspunkt zurückkehren. Die Fahrt von Stadt S_i nach S_j kostet dabei c_{ij}.
Ziel: Auswahl eines Wegs mit den geringsten Fahrtkosten.

Ein *Optimierungsproblem* besteht aus:

1. Menge L der zulässigen Lösungen. L wird häufig durch funktionale Gleichungen oder Ungleichungen und gegebenenfalls Ganzzahligkeitsbedingungen angegeben.

2. Reellwertige Zielfunktion z, die auf der Menge L minimiert oder maximiert werden soll. D.h. gesucht ist

$$x \in L : z(x) \leq z(y) \text{ bzw. } z(x) \geq z(y) \text{ für alle } y \in L$$

Lineare Optimierung (Linear Programming):
Die Zielfunktion z ist dabei eine lineare Funktion; L wird beschrieben durch lineare Gleichungen und Ungleichungen. Die Lösungen linearer Optimierungsprobleme sind im Allgemeinen nicht ganzzahlig. Lineare Optimierung liegt in P.

Ganzzahlige Optimierung (Integer Programming):
Unter „Integer Programming“ versteht man die Optimierung unter der Bedingung der Ganzzahligkeit der Lösungen. „Integer Programming“ ist NP-hart.

Obwohl sich fast alle diskreten Optimierungsprobleme als „Integer Programming“-Probleme formulieren und dann mit entsprechenden Algorithmen behandeln lassen, ist dies (aufgrund der Komplexität von „Integer Programming“) oft nicht praktikabel. Daher werden für viele wichtige Optimierungsprobleme andere Methoden verwandt, die unter dem Begriff „*kombinatorische Optimierung*“ firmieren. Häufig lassen sich solche Probleme graphentheoretisch formulieren und mit speziellen graphenteoretischen Optimierungsalgorithmen behandeln.

Kapitel 2

Lineare Optimierung

$f : \mathbb{R}^n \to \mathbb{R}$ linear, d.h. $f(x_1, \ldots, x_n) = c_1 x_1 + \cdots + c_n x_n$
Lineare Gleichung: $f(x_1, \ldots, x_n) = d$
Lineare Ungleichung: $f(x_1, \ldots, x_n) \leq d$ bzw. $f(x_1, \ldots, x_n) \geq d$
Lineare Gleichungen und lineare Ungleichungen werden im Folgenden als *lineare Restriktionen* bezeichnet.
Unter einem linearen Optimierungsproblem versteht man das Problem der Maximierung oder Minimierung einer linearen Zielfunktion unter Einhaltung von linearen Restriktionen.

2.1 Standardprogramme

Jedes lineare Programmierungsproblem lässt sich durch folgende Transformationen in die Form eines „Standardprogramms“ (siehe Definition 2.1) bringen:

1. Umkehren einer Ungleichung durch Multiplikation mit -1.

2. Ersetzung einer Gleichung $f(x_1, \ldots, x_n) = d$ durch die beiden Ungleichungen $f(x_1, \ldots, x_n) \leq d$ und $f(x_1, \ldots, x_n) \geq d$.

3. Ersetzung nichtrestringierter Variablen x_j (d.h. es wird weder $x_j \leq 0$ noch $x_j \geq 0$ gefordert) durch $x_j = x'_j - x''_j$ mit den Restriktionen $x'_j \geq 0$ und $x''_j \geq 0$.

Für das Folgende legen wir folgende Bezeichnungen fest: Sind $u = (u_1, \ldots, u_m)^T$, $v = (v_1, \ldots, v_m)^T \in \mathbb{R}^m$ so bedeutet $u \leq v$, dass $u_i \leq v_i$ für alle $i = 1, \ldots, m$.

2.1 Definition
Gegeben sei

$$A \in \mathbb{R}^{m\times n},\ b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} \in \mathbb{R}^m,\ c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \in \mathbb{R}^n.$$

Ein *Standard-Maximum-Programm* besteht in der Aufgabe, ein $x \in \mathbb{R}^n$ zu finden mit

$$\begin{array}{ll} (1) & Ax \le b,\ x \ge 0 \\ (2) & z(x) := c^T x \text{ maximal} \end{array}$$

(Bedingung (1) ist äquivalent zu

$$\left(\frac{A}{-E_n}\right) x \le \left(\frac{b}{0}\right))$$

Ein *Standard-Minimum-Programm* besteht in der Aufgabe, ein $x \in \mathbb{R}^n$ zu finden mit

$$\begin{array}{ll} (1) & Ax \ge b,\ x \ge 0 \\ (2) & z(x) := c^T x \text{ minimal} \end{array}$$

(Bedingung (1) ist in diesem Fall äquivalent zu

$$\left(\frac{A}{E_n}\right) x \ge \left(\frac{b}{0}\right))$$

Ein $x^* \in \mathbb{R}^n$, das (1) erfüllt, heißt *zulässige Lösung*. Eine zulässige Lösung x^* heißt *optimal*, falls die Zielfunktion maximiert bzw. minimiert wird.

Beachte: $c^T x$ maximal $\Leftrightarrow -c^T x$ minimal. D.h. das Standard-Maximum-Problem ist äquivalent zum Standard-Minimum-Problem.

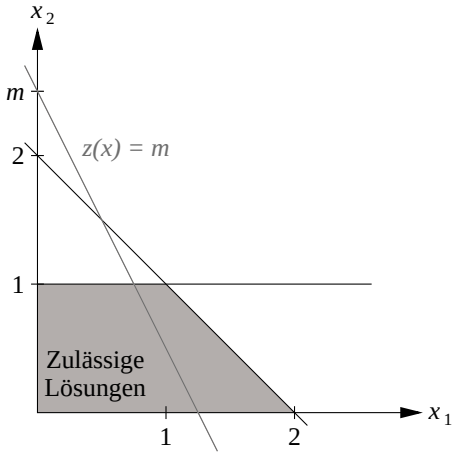
Beispiel 2.1:
Maximiere die Zielfunktion $z(x) = 2x_1 + x_2$ unter den folgenden linearen Restriktionen:

$$x_2 \le 1,\ x_1 + x_2 \le 2,\ x_1 \ge 0,\ x_2 \ge 0$$

Das Problem ist in Standard-Maximum-Form mit

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix},\ b = \begin{pmatrix} 1 \\ 2 \end{pmatrix},\ z(x) = c^T x \text{ mit } c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Grafische Darstellung des Problems:



Durch Paralellverschiebung der Geraden $2x_1 + x_2 = m$, sodass m maximal wird und die Gerade einen nicht-leeren Schnitt mit der Menge der zulässigen Lösungen hat, erhält man die Lösung des Optimierungsproblems: $x_1^* = 2$, $x_2^* = 0$, $z(x^*) = 4$.

Folgende Fragen werden im Folgenden beantwortet:
Gegeben sei ein Standardprogramm.

1. Wann existiert eine zulässige Lösung? Wann existiert eine optimale Lösung?
2. Wie sieht die Menge der zulässigen/optimalen Lösungen aus?
3. Wie berechnet man eine zulässige/optimale Lösung?

2.2 Dualität

Zu jedem Standardprogramm gehört ein duales Standardprogramm.

2.2 Definition
Sei

$$\text{(I)} \quad \begin{array}{l} Ax \leq b,\ x \geq 0 \\ z(x) := c^T x \text{ maximal} \end{array}$$

ein Standard-Maximum-Programm. (I) heißt *primales Programm.*
Das Standard-Minimum-Programm

$$\text{(I}^*\text{)} \quad \begin{array}{l} A^T y \geq c,\ y \geq 0\ (y \in \mathbb{R}^m) \\ z^*(y) := b^T y \text{ minimal} \end{array}$$

heißt das zu (I) *duale Programm.*
Umgekehrt heißt (I) das duale Programm zu (I*): (I) = (I**)

BEISPIEL 2.2:
Ein Standard-Maximum-Programm sei gegeben durch die Restriktionen

$$x_2 \leq 1,\ x_1 + x_2 \leq 2,\ x_1 \geq 0,\ x_2 \geq 0,$$

sowie die Zielfunktion $z(x) = 2x_1 + x_2$, die maximiert werden soll (vgl. Beispiel 2.1).
Damit ergeben sich für das primale Programm (I) folgende Koeffizienten:

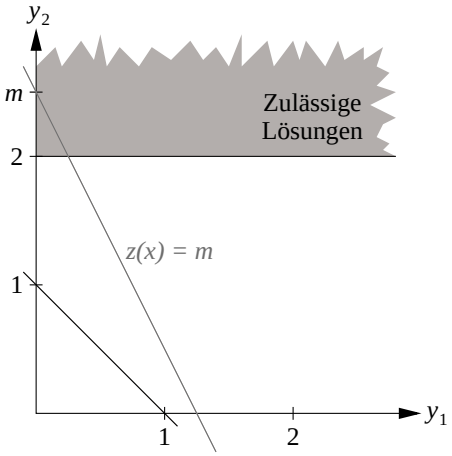
$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix},\ b = \begin{pmatrix} 1 \\ 2 \end{pmatrix},\ c = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Das duale Programm (I*) zu (I) hat somit folgende Form:

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \geq \begin{pmatrix} 2 \\ 1 \end{pmatrix},\ y \geq 0,\ \text{wobei } y \in \mathbb{R}^2$$

$$z^*(y) = y_1 + 2y_2 \text{ minimieren}$$

Grafische Darstellung des dualen Programms:



Die optimale Lösung des dualen Systems ist gegeben durch $y_1^* = 0$, $y_2^* = 2$, wobei $z^*(y^*) = 4$.
Beachte, dass in Beispiel 2.1 derselbe optimale Wert der Zielfunktion erhalten wurde. Dies ist kein Zufall, wie wir jetzt zeigen werden.

2.3 Satz
Seien x, y zulässige Lösungen von (I) bzw. (I*). Dann gilt:

$$z(x) = c^T x \leq y^T A x \leq b^T y = z^*(y)$$

BEWEIS.
Da x,y zulässige Lösungen für (I) bzw. (I*) sind, gilt: $x, y \geq 0$
Außerdem gilt nach Definition 2.2:

$$A^T y \geq c \ \Leftrightarrow\ (A^T y)^T \geq c^T \ \Leftrightarrow\ y^T A \geq c^T$$

Damit folgt:

$$c^T x \leq (y^T A)x = y^T(Ax) \leq y^T b = b^T y$$ ∎

2.4 Korollar
Seien x, y zulässige Lösungen von (I) bzw. (I*).
Gilt $c^T x = b^T y$, so ist x eine optimale Lösung von (I) und y eine optimale Lösung von (I*).

BEWEIS.
Zu zeigen ist, dass für jede beliebige, zulässige Lösung $\widetilde{x}$ von (I) $z(\widetilde{x}) \leq z(x)$ ist.
Nach Satz 2.3 gilt:

$$z(\widetilde{x}) = c^T \widetilde{x} \leq b^T y$$

Mit der Voraussetzung $b^T y = c^T x = z(x)$ ergibt sich dann daraus:

$$z(\widetilde{x}) \leq z(x)$$

Analog für y. ∎

2.3 Alternativsätze

Bezeichnungen
Ist A eine $m \times n$-Matrix über $\mathbb{R}$, dann bezeichnen im Folgenden $a^1, \ldots, a^n$ die Spaltenvektoren und $a_1^T, \ldots, a_m^T$ die Zeilenvektoren der Matrix A.

2.5 Lemma
Genau eine der beiden folgenden Möglichkeiten trifft zu:
(A) $Ax = b$ ist lösbar
(B) $A^T y = 0,\ b^T y = -1$ ist lösbar

BEWEIS.
Zwei Schritte:

1. Zeige, dass (A) und (B) nicht zugleich lösbar sind.
   Angenommen (A) und (B) sind zugleich lösbar mit x bzw. y. Damit ergibt sich:
   $$0 = x^T(A^T y) = (Ax)^T y = b^T y = -1 \qquad \text{Widerspruch!}$$

2. Zeige, dass wenn (A) nicht lösbar ist, dass dann (B) lösbar ist.
   Angenommen (A) ist nicht lösbar, d.h. b lässt sich nicht als Linearkombination der Spaltenvektoren $a^1, \ldots, a^n$ schreiben.

   Abbildung 2.1 zeigt die geometrische Begründung für die Existenz eines $y \in \mathbb{R}^m$, das (B) erfüllt: Der von $a^1, \ldots, a^n$ erzeugte Unterraum U (hier als Ebene gezeichnet) enthält nicht b, ist also ein echter Unterraum von R^m. Daher existiert ein Vektor $y \in \mathbb{R}^m$, der senkrecht auf U steht (d.h. $A^T y = 0$) und einen stumpfen Winkel ϕ mit b einschließt (wähle y auf anderer „Seite“ von U wie b). Wegen $\cos\phi = b^T y / |b||y|$ ist $b^T y < 0$. Durch

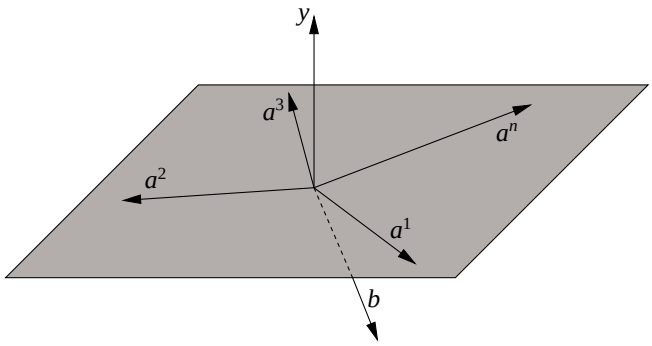


Abbildung 2.1: Grafische Darstellung der Aussage von Lemma 2.5 in drei Dimensionen

Multiplikation von y mit einer geeigneten reellen Zahl, kann man $b^T y = -1$ erreichen!

Algebraisch verläuft der Beweis folgendermaßen:
Ist $r(A)$ der Rang[1] von A, dann gilt wegen $b \notin \langle a^1, \ldots, a^n \rangle$:

$$r(A|b) = r(A) + 1$$

Dabei bezeichnet $A|b$ die $m \times (n+1)$-Matrix, die aus A durch Anfügen des Vektors b entstanden ist.
Sei nun $A' \in \mathbb{R}^{(m+1)\times(n+1)}$ mit

$$A' = \left(\begin{array}{c|c} A & b \\ \hline 0^T & -1 \end{array}\right)$$

Aus der Betrachtung des Spaltenrangs von A' ergibt sich $r(A') = r(A|b)$. Daraus wiederum lässt sich schlußfolgern, dass die Zeile $(0^T, -1)$ linear von den Zeilen der Matrix $(A|b)$ abhängig ist. D.h. es existieren $y_1, \ldots, y_m \in \mathbb{R}$ mit

$$\sum_{i=1}^m y_i a_i^T = 0,\ \sum_{i=1}^m y_i b_i = -1,\ \text{d.h.}$$

$$A^T y = 0,\ b^T y = -1$$

Also ist y Lösung von (B). ∎

2.6 Satz (Lemma von Farkas)
Genau eine der beiden folgenden Möglichkeiten trifft zu:

(A) $Ax = b,\ x \geq 0$ ist lösbar
(B) $A^T y \geq 0,\ b^T y < 0$ ist lösbar

[1] Als Zeilen- bzw. Spaltenrang einer Matrix A wird die maximale Anzahl der linear unabhängigen Zeilen bzw. Spalten bezeichnet. Ein Satz der linearen Algebra besagt, dass Zeilen- und Spaltenrang stets den selben Wert haben.

BEWEIS.
Zwei Schritte:

1. Zeige, dass (A) und (B) nicht zugleich lösbar sind.
   Angenommen (A) und (B) sind zugleich lösbar mit x bzw. y. Da nach Voraussetzung $x^T \geq 0$ und $A^T y \geq 0$ ergibt sich:

   $$0 \leq x^T(A^T y) = (Ax)^T y \leq b^T y < 0 \qquad \text{Widerspruch!}$$

2. Zeige, dass wenn (A) nicht lösbar ist, dass dann (B) lösbar ist.
   Angenommen (A) ist nicht lösbar.
   Falls $Ax = b$ nicht lösbar ist, sind wir fertig mit Lemma 2.5.
   Wir können also annehmen, dass $Ax = b$ lösbar ist, aber keine nicht-negativen Lösungen besitzt. Insbesondere ist $b \neq 0$.

   Wir beweisen die Lösbarkeit von (B) durch Induktion nach n, der Spaltenanzahl von A:

   Induktionsanfang: $n = 1$
   Dann ist $A = a^1$. Sei $x_1 a^1 = b$, $x_1 < 0$ (beachte: x_1 ist eine reelle Zahl).
   Setze $y = -b$. Dann gilt:

   $$A^T y = (a^1)^T y = -\frac{1}{|x_1|} b^T y = \frac{1}{|x_1|} b^T b > 0 \text{ und } b^T y = -b^T b < 0$$

   Also ist y Lösung von (B).

   Induktionsschluss: Behauptung sei richtig für $n-1$. Zu zeigen ist, dass die Behauptung gilt für n.
   Da (A) keine nicht-negative Lösung hat, gilt dies auch für die Gleichung

   $$\sum_{i=1}^{n-1} x_i a^i = b$$

   Nach Induktionsvoraussetzung existiert $v \in \mathbb{R}^m$ mit $(a^i)^T v \geq 0$, $i = 1, \ldots, n-1$ und $b^T v < 0$. Ist auch $(a^n)^T v \geq 0$, so ist v eine Lösung von (B) und wir sind fertig.
   Sei also $(a^n)^T v < 0$. Wir setzen nun:

   $$\begin{aligned} \overline{a}^i &= \left((a^i)^T v\right) a^n - \left((a^n)^T v\right) a^i, \; i = 1, \ldots, n-1 \\ \overline{b} &= \left(b^T v\right) a^n - \left((a^n)^T v\right) b \end{aligned}$$

   Zwischenbehauptung: $\sum_{i=1}^{n-1} x_i \overline{a}^i = \overline{b}$, $x_i \geq 0$, $i = 1, \ldots, n-1$ ist nicht lösbar.
   Angenommen doch. Dann ist

   $$\sum_{i=1}^{n-1} x_i \left[\left((a^i)^T v\right) a^n - \left((a^n)^T v\right) a^i\right] = (b^T v) a^n - \left((a^n)^T v\right) b.$$

Also

$$\underbrace{-\frac{1}{(a^n)^T v}\left[\sum_{i=1}^{n-1} x_i\left((a^i)^T v\right) - b^T v\right]}_{\text{Koeffizient von } a^n \geq 0} a^n + \sum_{i=1}^{n-1} x_i a^i = b$$

Da die Koeffizienten von a^n und den a^i auf der linken Seite dieser Gleichung nicht-negativ sind, hat man damit eine Lösung von (A). Mit diesem Widerspruch ist somit die Zwischenbehauptung bewiesen.

Wegen der Zwischenbehauptung existiert nach Induktionsvoraussetzung ein $w \in \mathbb{R}^m$ mit $(\overline{a}^i)^T w \geq 0$ für $i = 1, \ldots, n-1$ und $\overline{b}^T w < 0$. Dann ist aber $y = \left((a^n)^T w\right) v - \left((a^n)^T v\right) w$ eine Lösung von (B), denn:

$$\begin{aligned}
(a^i)^T y &= \left((a^n)^T w\right)\left((a^i)^T v\right) - \left((a^n)^T v\right)\left((a^i)^T w\right) \\
&= (\overline{a}^i)^T w \geq 0,\ i = 1, \ldots, n-1
\end{aligned}$$

$$\begin{aligned}
(a^n)^T y &= \left((a^n)^T w\right)\left((a^n)^T v\right) - \left((a^n)^T v\right)\left((a^n)^T w\right) \\
&= 0
\end{aligned}$$

$$\begin{aligned}
b^T y &= \left((a^n)^T w\right)\left(b^T v\right) - \left((a^n)^T v\right)\left(b^T w\right) \\
&= \overline{b}^T w < 0
\end{aligned}$$

■

2.7 Korollar
Genau eine der beiden folgenden Möglichkeiten trifft zu:

(A) $Ax \leq b,\ x \geq 0$ ist lösbar
(B) $A^T y \geq 0,\ y \geq 0,\ b^T y < 0$ ist lösbar

BEWEIS.
Zwei Schritte:

1. Zeige, dass (A) und (B) nicht zugleich lösbar sind.
   Angenommen (A) und (B) sind zugleich lösbar mit x bzw. y. Da nach Voraussetzung $x^T \geq 0$ und $A^T y \geq 0$ ergibt sich:

   $$0 \leq x^T (A^T y) = (Ax)^T y \leq b^T y < 0 \qquad \text{Widerspruch!}$$

2. Zeige, dass wenn (A) nicht lösbar ist, dass dann auf jeden Fall (B) lösbar ist.
   Angenommen (A) ist nicht lösbar. Dann besitzt auch

   $$Ax + z = b,\ x \geq 0,\ z \geq 0 \qquad (*)$$

   keine Lösung.
   Setze $B = (A|E_m)$, wobei E_m die Einheitsmatrix mit m Zeilen bzw. Spalten ist. Damit lässt sich dann $(*)$ schreiben als

   $$\underbrace{Bw = b,\ w \geq 0}_{\text{entspricht (A) in Satz 2.6}} \quad , \text{ wobei } w = \begin{pmatrix} x \\ z \end{pmatrix}$$

Nach Satz 2.6 existiert $y \in \mathbb{R}^m$ mit $B^T y \geq 0$ und $b^T y < 0$
Daraus folgt $A^T y \geq 0$ und $y = E_m y \geq 0$. Also ist (B) lösbar. ∎

2.4 Der Hauptsatz der linearen Optimierung

2.8 Satz (Hauptsatz der linearen Optimierung)
Gegeben sei ein Standardprogramm (I) und das zugehörige duale Programm (I*) (Bezeichnungen wie in Definition 2.2). Dann gilt:

a) Haben (I) und (I*) beide zulässige Lösungen, dann haben auch beide optimale Lösungen. Ist x^* optimale Lösung von (I) und y^* optimale Lösung von (I*), so ist
$$z(x^*) = z^*(y^*)$$

b) Besitzt eines von (I) bzw. (I*) keine zulässigen Lösungen, so hat *keines* der beiden Programme eine optimale Lösung.

c) Besitzt (I*) zulässige Lösungen und (I) nicht, so ist z^* auf der Menge der zulässigen Lösungen von (I*) nicht nach unten beschränkt.
Besitzt umgekehrt (I) zulässige Lösungen und (I*) nicht, so ist z auf der Menge der zulässigen Lösungen von (I) nicht nach oben beschränkt.

Beweis.
Beweis für a):
Aus Satz 2.3 und Korollar 2.4 folgt die Behauptung, wenn wir zeigen können, dass das System
$$\begin{array}{rcll} Ax & \leq & b, & x \geq 0 \\ A^T y & \geq & c, & y \geq 0 \\ c^T x - b^T y & \geq & 0 & \end{array}$$
eine Lösung $\begin{pmatrix} x^* \\ y^* \end{pmatrix}$ besitzt. Dann sind x^* und y^* optimale Lösungen von (I) bzw. (I*) und $z(x^*) = z^*(y^*)$.
Durch Umschreiben des Systems erhält man:
$$(*) \quad \underbrace{\begin{pmatrix} A & 0 \\ 0 & -A^T \\ -c^T & b^T \end{pmatrix}}_{(n+m+1)\times(n+m)} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} b \\ -c \\ 0 \end{pmatrix}, \ \begin{pmatrix} x \\ y \end{pmatrix} \geq 0$$

Angenommen $(*)$ hat keine Lösung. Nach Korollar 2.7 existieren dann $z \in \mathbb{R}^m$, $w \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$ mit
$$\begin{pmatrix} A^T & 0 & -c \\ 0 & -A & b \end{pmatrix} \begin{pmatrix} z \\ w \\ \alpha \end{pmatrix} \geq 0, \ \begin{pmatrix} z \\ w \\ \alpha \end{pmatrix} \geq 0, \ \begin{pmatrix} b \\ -c \\ 0 \end{pmatrix}^T \begin{pmatrix} z \\ w \\ \alpha \end{pmatrix} < 0$$

Durch Umschreiben erhält man folgendes System:

$$\begin{array}{rrcll} & A^T z & \geq & \alpha c, & z \geq 0,\ \alpha \geq 0 \\ & Aw & \leq & \alpha b, & w \geq 0 \\ (**) & b^T z & < & c^T w & \end{array}$$

Wir zeigen, dass $\alpha \neq 0$.
Angenommen $\alpha = 0$. Dann ist $A^T z \geq 0$, $Aw \leq 0$. Sind $\overline{x}$ und $\overline{y}$ zulässige Lösungen von (I) bzw. (I*), dann gilt:

$$\begin{array}{l} A\overline{x} \leq b,\ \overline{x} \geq 0 \\ A^T\overline{y} \geq c,\ \overline{y} \geq 0 \end{array}$$

Damit ergibt sich dann:

$$0 \leq \overline{x}^T(A^T z) = (A\overline{x})^T z \leq b^T z < c^T w \leq (A^T\overline{y})^T w = \overline{y}^T A w \leq 0$$

Widerspruch ($0 < 0$)! Also ist $\alpha > 0$.
Sei $x = \frac{w}{\alpha}$, $y = \frac{z}{\alpha}$. Dann gilt:

$$\begin{array}{lll} Ax \leq b & (\text{ da } Aw \leq \alpha b), & x \geq 0 \\ A^T y \geq c & (\text{ da } A^T z \geq \alpha c), & y \geq 0 \end{array}$$

x und y sind also zulässige Lösungen von (I) bzw. (I*). Mit Satz 2.3 folgt damit:

$$c^T w = \alpha(c^T x) \leq \alpha(b^T y) = b^T z$$

Widerspruch zu $(**)$! Also hat $(*)$ eine Lösung und a) ist bewiesen.

Beweis für b) und c):
Angenommen (I) hat keine zulässigen Lösungen und somit auch keine optimale Lösung. Nach Korollar 2.7 existiert dann ein $w \in \mathbb{R}^m$ mit $A^T w \geq 0$, $w \geq 0$, $b^T w < 0$.
Falls (I*) überhaupt eine zulässige Lösung $\overline{y}$ besitzt, so ist auch $\overline{y} + \lambda w$ für alle $\lambda \geq 0$ eine zulässige Lösung von (I*), denn:

$$A^T \underbrace{(\overline{y} + \lambda w)}_{\geq 0} = \underbrace{A^T\overline{y}}_{\geq c} + \underbrace{\lambda A^T w}_{\geq 0} \geq c$$

Die Zielfunktion z^* von (I*) nimmt für diese zulässigen Lösungen folgende Werte an:

$$z^*(\overline{y} + \lambda w) = b^T\overline{y} + \lambda \underbrace{b^T w}_{<0}$$

z^* ist also auf der Menge der zulässigen Lösungen nicht nach unten beschränkt ($\overline{y}$ und w sind fest. Wird λ immer größer, so wird z^* immer kleiner.). Daraus folgt dann wiederum, dass z^* auf der Menge der zulässigen Lösungen von (I*) kein Minimum besitzt.
Analog schließt man, wenn (I*) keine zulässige Lösung hat. ∎

2.9 Korollar
Gegeben sei ein Standardprogramm (I) und das zugehörige duale Programm (I^*). Dann gilt:
(I) besitzt genau dann eine optimale Lösung, wenn (I^*) eine optimale Lösung besitzt.

2.10 Korollar
Ein Standard-Maximum-Programm (Standard-Minimum-Programm) besitzt genau dann eine optimale Lösung, wenn es zulässige Lösungen gibt und die Zielfunktion auf der Menge der zulässigen Lösungen nach oben (unten) beschränkt ist.

BEWEIS.
Sei (I) ein Standard-Maximum-Programm, welches zulässige Lösungen besitzt und dessen Zielfunktion z auf der Menge dieser zulässigen Lösungen nach oben beschränkt ist.
Angenommen das zugehörige duale Programm (I^*) hat keine zulässigen Lösungen. Dann ist nach Satz 2.8c) die Zielfunktion z von (I) auf der Menge der zulässigen Lösungen nicht nach oben beschränkt. Widerspruch!
Also besitzt auch (I^*) zulässige Lösungen. Somit folgt nach Satz 2.8a), dass (I) eine optimale Lösung besitzt. ∎

2.11 Bemerkung
Nach Satz 2.8 und Korollar 2.10 gilt:
Ein Standard-Maximum-Programm (Standard-Minimum-Programm) erfüllt genau einen der drei folgenden Fälle:

1. Das Programm hat eine optimale Lösung.
2. Das Programm hat zulässige Lösungen, wobei die Zielfunktion auf der Menge dieser zulässigen Lösungen nicht nach oben (unten) beschränkt ist.
3. Das Programm hat keine zulässigen Lösungen.

Sei (I) das primale und (I^*) das zugehörige duale Programm. Tabelle 2.1 zeigt, welche Kombinationen der drei Fälle möglich bzw. nicht möglich sind.

		primales Programm (I)		
		1.	2.	3.
duales Programm (I^*)	1.	*möglich* (vgl. Beispiel 2.1 und 2.2)	*nicht möglich* (wegen Korollar 2.9)	*nicht möglich* (wegen Korollar 2.9)
	2.	*nicht möglich* (wegen Korollar 2.9)	*nicht möglich* (wegen Satz 2.8)	*möglich* (vgl. Beispiel 2.3a))
	3.	*nicht möglich* (wegen Korollar 2.9)	*möglich* (vgl. Beispiel 2.3a))	*möglich* (vgl. Beispiel 2.3b))

Tabelle 2.1: Kombination verschiedener Lösungsmöglichkeiten (vgl. Bemerkung 2.11)

BEISPIEL 2.3:
Diese beiden Beispiele sollen dienen der Veranschaulichung der in Tabelle 2.1 dargestellten Kombinationsmöglichkeiten.

a) (I) $x \leq -1$, $x \geq 0$, $z(x) = x$ maximal
Besitzt keine zulässigen Lösungen.

(I*) $y \geq 1$, $y \geq 0$, $z^*(y) = -y$ minimal
Besitzt zulässige Lösungen. z^* besitzt jedoch kein Minimum.

b) (I) $x_1 + x_2 \leq 2$, $x_1 - x_2 \leq 1$, $x_1 \geq 0,\ x_2 \geq 0$, $z(x) = -x_1 + 2x_2$ maximal
Besitzt keine zulässigen Lösungen.

(I*) $-y_1 + y_2 \geq -1$, $y_1 - y_2 \geq 2$, $y_1 \geq 0,\ y_2 \geq 0$, $z^*(y) = -2y_1 + y_2$ minimal
Besitzt keine zulässigen Lösungen.

2.5 Zulässige und optimale Lösungen

Wir benötigen jetzt eine andere Formulierung von linearen Optimierungsproblemen.

2.12 Definition
Gegeben $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.
Ein *kanonisches Maximum-Programm* besteht in der Aufgabe ein $x \in \mathbb{R}^n$ zu finden mit:

$$Ax = b,\ x \geq 0$$
$$z(x) = c^T x \text{ maximal}$$

Ein *kanonisches Minimum-Programm* besteht in der Aufgabe ein $x \in \mathbb{R}^n$ zu finden mit:

$$Ax = b,\ x \geq 0$$
$$z(x) = c^T x \text{ minimal}$$

2.13 Bemerkung (Standard-Programm $\leftrightarrow$ kanonisches Programm)
Ein kanonisches Maximum-Programm der Form

$$Ax = b,\ x \geq 0$$
$$z(x) = c^T x \text{ maximal}$$

lässt sich durch Umschreiben der Gleichungen in zwei Ungleichungen in ein äquivalentes Standard-Maximum-Programm umformen, welches folgende Form besitzt:

$$Ax \leq b,\ -Ax \leq -b,\ x \geq 0$$
$$c^T x \text{ maximal}$$

Analog lässt sich ein Standard-Maximum-Programm der Form

$$Ax \leq b,\ x \geq 0$$
$$c^T x \text{ maximal}$$

durch die Einführung zusätzlicher Variablen $(z_1, \ldots, z_m) = z^T$ (sog. *Schlupfvariablen*) in ein äquivalentes kanonisches Maximum-Programm umformen, welches folgende Form besitzt:

$$\begin{array}{c} Ax + z = b,\ x \geq 0,\ z \geq 0 \\ c^T x + 0^T z \text{ maximal} \end{array} \quad \Leftrightarrow \quad \begin{array}{c} (A|E_m) \begin{pmatrix} x \\ z \end{pmatrix} = b,\ \begin{pmatrix} x \\ z \end{pmatrix} \geq 0 \\ (c^T, 0^T) \begin{pmatrix} x \\ z \end{pmatrix} \text{ maximal} \end{array}$$

Wir betrachten im Folgenden das kanononische Minimum-Programm

$$\begin{array}{ll} \text{(I)} & Ax = b,\ x \geq 0 \\ & z(x) = c^T x \text{ minimal} \end{array}$$

Jedes kanonische Maximum-Programm lässt sich durch Ersetzung von c durch $-c$ in ein kanonisches Minimum-Programm umwandeln.
Wir können außerdem annehmen, dass $r(A) = r(A|b) = m \leq n$.
Im Fall $r(A) < r(A|b)$ besitzt das Gleichungssystem $Ax = b$ keine Lösung.
Im Fall, dass $r(A) = r(A|b) < m$ ist, lassen sich $m - r(A)$ viele Gleichungen als Linearkombination von geeigneten $r(A)$ vielen ausdrücken. Dann gilt auf jeden Fall $m \leq n$, da $\dim\mathbb{R}^n = n$.
Die Menge aller zulässigen Lösungen von (I) heißt L. Die Menge der optimalen Lösungen heißt L_{opt}.

2.14 Definition

a) $K \subseteq \mathbb{R}^n$ heißt *konvex*, falls für alle $x, x' \in K$ und alle $0 \leq \lambda \leq 1$ gilt:

$$\lambda x + (1 - \lambda)x' \in K$$

(d.h. die Verbindungsstrecke zwischen x und x' liegt vollständig in K).

b) Sei $K \subseteq \mathbb{R}^n$ konvex. $p \in K$ heißt *Ecke* von K, falls p nicht im Inneren einer ganz in K liegenden Strecke liegt. D.h. $p = \lambda x + (1-\lambda)x'$, $x, x' \in K$, $0 \leq \lambda \leq 1$, dann ist $\lambda = 0$ $(p = x')$ oder $\lambda = 1$ $(p = x)$.

Mit $E(K)$ wird im Folgenden die Menge der Ecken von K bezeichnet. Dabei können folgende Fälle auftreten:

- $E(K) = \emptyset$ (z.B. wenn $K = \mathbb{R}^n$)
- $E(K)$ endlich
- $E(K)$ unendlich (z.B. wenn $K \subset \mathbb{R}^2$ ein Kreis ist)

2.15 Satz

Die Menge L der zulässigen Lösungen von (I) ist konvex und abgeschlossen[2].

[2]Eine Teilmenge A eines mit einer Norm versehenen Vektorraums V heißt abgeschlossen, wenn mit jeder konvergenten Folge, deren Glieder in A liegen, auch der Grenzwert zu A gehört: $u_k \to u,\ u_k \in A \Rightarrow u \in A$

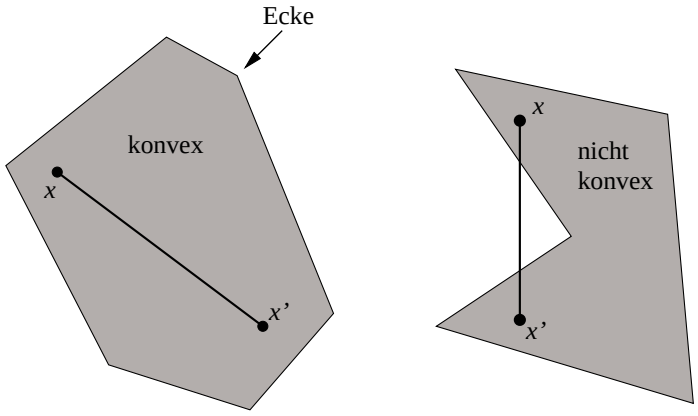


Abbildung 2.2: Konvexe und nicht konvexe Teilmenge des $\mathbb{R}^2$

BEWEIS.
$a_i^T x = b_i$ (m Gleichungen)
Die Lösungemenge H_i jeder dieser Gleichungen ist eine affine Hyperebene, also konvex und abgeschlossen. Dies lässt sich leicht wie folgt zeigen:

1. Konvexität
   x', x'' seien Lösungen von $a_i^T x = b_i$. Dann ist für $0 \leq \lambda \leq 1$ auch $x''' = \lambda x' + (1-\lambda)x''$ eine Lösung von $a_i^T x = b_i$, denn

   $$a_i^T(\lambda x' + (1-\lambda)x'') = \lambda a_i^T x' + (1-\lambda)a_i^T x'' = \lambda b_i + (1-\lambda)b_i = b_i$$

   Also ist die Lösungsmenge von $a_i^T x = b_i$ konvex.

2. Abgeschlossenheit
   $x \mapsto a_i^T x$ ist stetig.
   Seien x_j Lösungen von $a_i^T x = b_i$. Konvergiert die Folge der x_j gegen x ($x_j \to x$), dann konvergiert die Folge $a_i^T x_j$ gegen $a_i^T x$:

   $$\underbrace{a_i^T x_j}_{=b_i} \to a_i^T x$$

   D.h. $a_i^T x = b_i$. Die Menge der Lösungen von $a_i^T x = b_i$ ist also abgeschlossen.

Ebenso sieht man, dass die Mengen $P_j = \{x \in \mathbb{R}^n | x_j \geq 0\}$ konvex und abgeschlossen sind.
Da der Schnitt konvexer/abgeschlossener Mengen wieder konvex/abgeschlossen ist gilt:

$$L = \bigcap_{i=1}^{m} H_i \cap \bigcap_{j=1}^{n} P_j \text{ ist konvex und abgeschlossen.}$$

■

2.16 Definition
Sei L die Menge der zulässigen Lösungen des kanonischen Minimum-Programms

$$\begin{array}{ll}(\mathrm{I}) & Ax = b,\ x \geq 0 \\ & z(x) = c^T x \text{ minimal}\end{array}$$

Sei $x \in L$ und $Z \subseteq \{1, \ldots, n\}$ die Menge der Indizes k mit $x_k > 0$, also $x_j = 0$ für $j \in \{1, \ldots, n\} \setminus Z$.
Wir nennen $\{a^k \mid k \in Z\}$ die *zu x gehörende Spaltenmenge*. Somit gilt:

$$\sum_{k \in Z} a^k x = \sum_{j=1}^{n} a^j x = Ax = b$$

2.17 Satz
Sei die Menge L der zulässigen Lösungen des kanonischen Minimum-Programms

$$\begin{array}{ll}(\mathrm{I}) & Ax = b,\ x \geq 0 \\ & z(x) = c^T x \text{ minimal}\end{array}$$

nicht leer. Dann hat L Ecken.

Beweis.
Wähle unter allen $x \in L$ ein solches, für das die Menge Z der Indizes k mit $x_k > 0$ minimale Größe hat.
Ist $x \notin E(L)$, so existieren $x', x'' \in L$, $x' \neq x''$ und ein $0 < \lambda < 1$ mit $x = \lambda x' + (1-\lambda)x''$, also $x_j = \lambda x'_j + (1-\lambda)x''_j$ für alle $j = 1, \ldots, n$.
Ist $j \notin Z$, so ist $x_j = 0$, also $-\lambda x'_j = (1-\lambda)x''_j$. Da $x'_j, x''_j \geq 0$, $\lambda > 0$, $1-\lambda > 0$, folgt $x'_j = x''_j = 0$.
Folglich ist $\sum_{k \in Z} a^k x'_k = Ax' = b = Ax'' = \sum_{k \in Z} a^k x''_k$, d.h.

$$\sum_{k \in Z} (x'_k - x''_k)\, a^k = 0$$

Setze $v_k = x'_k - x''_k$, also $\sum_{k \in Z} v_k a^k = 0$ und mindestens ein $v_k \neq 0$.
Sei $\rho = \min\{x_k/|v_k| \mid k \in Z,\ v_k \neq 0\}$. Sei $h \in Z$ mit $\rho = x_h/|v_h|$. Indem wir gegebenenfalls x' und x'' vertauschen, können wir $v_h > 0$, also $\rho = x_h/v_h$ annehmen.
Setze $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_n)^T$ mit $\overline{x}_k = x_k - \rho v_k$ für $k \in Z$ und $\overline{x}_j = 0$ für $j \notin Z$. Dann gilt $\overline{x} \geq 0$ (denn $\rho v_k \leq x_k$, falls $v_k > 0$ nach Definition von ρ). Ferner ist $A\overline{x} = Ax - \rho \sum_{k \in Z} v_k a^k = Ax = b$. Also $\overline{x} \in L$.
Da $\overline{x}_h = 0$, ist die zu $\overline{x}$ gehörende Menge der Indizes k mit $\overline{x}_k > 0$ eine echte Teilmenge von Z. Dies ist ein Widerspruch zur Wahl von Z.
Daher ist $x \in E(L)$. ∎

Beispiel 2.4:
Gegeben sei ein kanonisches Minimum-Programm

$$\begin{array}{ll}(\mathrm{I}) & Ax = b,\ x \geq 0 \\ & z(x) = c^T x \text{ minimal}\end{array}$$

mit $A = (1\ 1\ 1)$ und $b = 1$.
Abbildung 2.3 zeigt eine dreieckige Fläche im $\mathbb{R}^3$, welche den zulässigen Lösungen L für dieses Problems entspricht. Man erkennt, dass diese Menge L Ecken besitzt. In diesem Fall sind das diejenigen Punkte der Lösungsmenge, bei denen zwei Koordinaten $= 0$ sind. Jeder andere Punkt von L hat mindestens zwei von 0 verschiedene Koordinaten. Man mache sich den Beweis des Satzes 2.17 an diesem Beispiel klar.

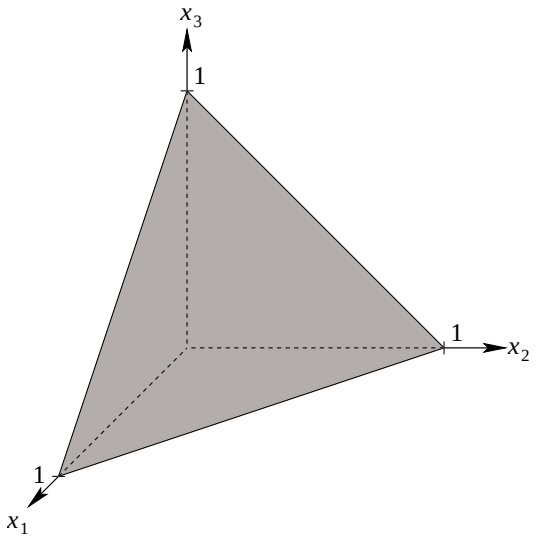


Abbildung 2.3: Zulässige Lösungen für $x_1 + x_2 + x_3 = 1,\ x_1, x_2, x_3 \geq 0$

2.18 Satz
Sei $x \in L$. Dann gilt:
x ist genau dann eine Ecke von L, wenn die zu x gehörende Spaltenmenge linear unabhängig ist.

Beweis.
Sei $\{a^k \mid k \in Z\}$ die zu x gehörende Spaltenmenge.
Zwei Richtungen:

„$\Leftarrow$“ Ist x keine Ecke, so existieren $x', x'' \in L$, $x' \neq x''$, $0 < \lambda < 1$ mit $x = \lambda x' + (1-\lambda)x''$. D.h. $x_j = \lambda x'_j + (1-\lambda)x''_j$. Da $\lambda, 1-\lambda > 0$ und $x_j = 0$ für $j \notin Z$, gilt also:
$x'_j = x''_j = 0$ für $j \notin Z$.
Folglich ist

$$\sum_{k \in Z} x'_k a^k = \sum_{k=1}^{n} x'_k a^k = Ax' = b = Ax'' = \sum_{k=1}^{n} x''_k a^k = \sum_{k \in Z} x''_k a^k$$

Damit ergibt sich also:

$$\sum_{k\in Z}(x'_k - x''_k)a^k = 0$$

Also ist $\{a^k \mid k \in Z\}$ linear abhängig.

„⇒“ Ist die zu x gehörende Spaltenmenge linear abhängig, dann existieren $v_k \in \mathbb{R}$, $k \in Z$ mit mindestens einem $v_{k_0} \neq 0$, $k_0 \in Z$, sodass $\sum_{k\in Z} v_k a^k = 0$. Da $x_k > 0$ für $k \in Z$, gilt für genügend kleines $\rho > 0$, dass

$$x_k \pm \rho v_k > 0 \text{ für alle } k \in Z$$

Sei $x'_k = x_k + \rho v_k$, $x''_k = x_k - \rho v_k$ für $k \in Z$, $x'_j = x''_j = 0$ für $j \notin Z$. Dabei sind x_{k_0}, x'_{k_0}, x''_{k_0} paarweise verschieden, da $v_{k_0} \neq 0$ ist. Somit sind auch x, x', x'' paarweise verschieden mit $x', x'' \geq 0$.
Damit ist $x', x'' \in L$, denn

$$Ax' = \sum_{k\in Z} a^k x'_k = \underbrace{\sum_{k\in Z} a^k x_k}_{=b} + \rho \underbrace{\sum_{k\in Z} v_k a^k}_{=0} = b$$

$$Ax'' = \sum_{k\in Z} a^k x''_k = \underbrace{\sum_{k\in Z} a^k x_k}_{=b} - \rho \underbrace{\sum_{k\in Z} v_k a^k}_{=0} = b$$

Damit ergibt sich nun

$$x = \frac{1}{2}x' + \frac{1}{2}x'',$$

denn es gilt

$$\frac{1}{2}x'_k + \frac{1}{2}x''_k = \frac{1}{2}(x_k + \rho v_k) + \frac{1}{2}(x_k - \rho v_k) = x_k \text{ für } k \in Z$$

$$\frac{1}{2}x'_j + \frac{1}{2}x''_j = 0 = x_j \text{ für } j \notin Z$$

Also ist x keine Ecke. ∎

2.19 Korollar
Ist x eine Ecke von L, so hat x höchstens m viele positive Koordinaten.

Beweis.
Es gibt höchstend m viele linear unabhängige Spalten. Mit Satz 2.18 folgt direkt die Aussage. ∎

2.20 Korollar
Ist $L \neq \emptyset$, so hat L endlich viele Ecken.

BEWEIS.
Ist $a^{i_1}, \ldots, a^{i_r}$ eine linear unabhängige Teilmenge der Spalten von A, so existiert höchstens ein $x \in L$ mit zugehöriger Spaltenmenge $\{a^{i_1}, \ldots, a^{i_r}\}$ (gäbe es ein weiteres solches $x' \in L$, $x' \neq x$, so folgte $\sum_{j=1}^{r} a^{i_j}(x_{i_j} - x'_{i_j}) = 0$ und $a^{i_1}, \ldots, a^{i_r}$ wäre nicht linear unabhängig).
Nach Satz 2.18 gibt es also höchstens so viele Ecken in L, wie es linear unabhängige Teilmengen von Spalten von A gibt. Dies sind endlich viele. ∎

2.21 Satz
Die Menge L_{opt} ist konvex und abgeschlossen.
Es gilt: $E(L_{\text{opt}}) = E(L) \cap L_{\text{opt}}$.

BEWEIS.
Ist $L_{\text{opt}} = \emptyset$, so sind wir fertig, da die leere Menge vereinbarungsgemäß konvex und abgeschlossen ist.
Sei $L_{\text{opt}} \neq \emptyset$ und μ der Minimalwert von $c^T x$ auf L.
Dann ist $L_{\text{opt}} = L \cap \{x \in \mathbb{R}^n \mid c^T x = \mu\}$. L ist nach Satz 2.15 konvex und abgeschlossen. Ebenso ist $\{x \in \mathbb{R}^n \mid c^T x = \mu\}$ konvex und abgeschlossen. Folglich ist auch der Schnitt von beiden Mengen, d.h. also L_{opt} konvex und abgeschlossen.
Ist $x \in E(L) \cap L_{\text{opt}}$, so liegt x insbesondere nicht im Inneren einer Verbindungsstrecke zweier Punkte aus L_{opt}, d.h. $E(L) \cap L_{\text{opt}} \subseteq E(L_{\text{opt}})$.
Sei umgekehrt $x \in E(L_{\text{opt}})$ und seien $u, v \in L$, $u \neq v$ mit $x = \lambda u + (1-\lambda)v$, $0 \leq \lambda \leq 1$. Dann ist

$$\mu = c^T x = \lambda \underbrace{c^T u}_{\geq \mu} + (1-\lambda) \underbrace{c^T v}_{\geq \mu} \geq \lambda\mu + (1-\lambda)\mu = \mu$$

Also gilt Gleichheit, und es folgt $c^T u = \mu$ und $c^T v = \mu$. D.h. $u, v \in L_{\text{opt}}$. Da $x \in E(L_{\text{opt}})$, folgt $x = u$ oder $x = v$. Also $x \in E(L)$. ∎

2.22 Satz
Sei $L_{\text{opt}} \neq \emptyset$, so ist auch $E(L_{\text{opt}}) \neq \emptyset$. D.h. existieren Optimallösungen, so befindet sich mindestens eine von ihnen in einer Ecke.

BEWEIS.
Sei $L_{\text{opt}} \neq \emptyset$, μ der Minimalwert von $z(x) = c^T x$. Ergänzen wir nun $Ax = b$ durch die Gleichung $c^T x = \mu$, so erhalten wir folgendes Gleichungssystem:

$$\left(\frac{A}{c^T}\right) x = \left(\frac{b}{\mu}\right), \; x \geq 0 \qquad (*)$$

Die Menge der zulässigen Lösungen von $(*)$ ist genau L_{opt}. Nach Satz 2.17 enthält L_{opt} daher Ecken.
(*Anmerkung*: Hier geht die Linearität der Zielfunktion ein. Der Satz 2.22 gilt tatsächlich i.a. nicht für nicht-lineare Zielfunktionen. Wählt man z.B. in Beispiel 2.4 die Zielfunktion $z(x) = x_1 x_2 x_3$, die maximiert werden soll, so ist die optimale Lösung $\frac{1}{27}$, die genau bei $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)^T$ angenommen wird, also nicht an einer Ecke.) ∎

Bezeichnungen
Zur Beschreibung des Simplex-Algorithmus in Abschnitt 2.6 führen wir jetzt noch folgende Bezeichnungen ein:
Sei $x \in E(L)$. Jede Menge von m linear unabhängigen Spalten von A, die die zu x gehörende Spaltenmenge enthält, heißt *Basis* zu x. Jedes $x \in E(L)$ besitzt mindestens eine Basis (vgl. lineare Algebra).
Besitzt ein $x \in E(L)$ m viele Einträge ungleich 0, so heißt x eine *nicht-entartete Ecke*, ansonsten *entartete Ecke*. Nicht-entartete Ecken besitzen genau eine Basis, nämlich die zugehörige Spaltenmenge.

2.6 Simplex-Algorithmus

Gegeben sei ein kanonisches Minimum-Programm $(*)$ (Bezeichnungen wie in Definition 2.12).
Der Simplex-Algorithmus besteht aus zwei Teilen:

(I) Finde zulässige Ecke x^0.

(II) Stelle fest, ob x^0 optimal ist. Wenn nicht, finde $x^1 \in E(L)$ mit $z(x^1) < z(x^0)$ oder stelle fest, dass es keine optimale Lösung gibt.

Der Algorithmus terminiert nach endlich vielen Schritten.

Teil (II) des Algorithmus
Sei $x^0 = (x_1^0, \ldots, x_n^0)^T \in L$ eine Ecke, $Z \subseteq \{1, \ldots, n\}$ Menge der Indizes i mit $x_i^0 > 0$. Sei $\overline{Z} \supseteq Z$, $|\overline{Z}| = m$, so dass $\{a^s | s \in \overline{Z}\}$ Basis von x^0 ist.
Dann gilt für alle $j = 1, \ldots, n$

$$a^j = \sum_{s \in \overline{Z}} \tau_{sj} a^s, \text{ wobei } \tau_{sj} \in \mathbb{R} \text{ eindeutig bestimmt sind} \qquad \text{(II.1)}$$

Ist $s \in \overline{Z}$, so ist $\tau_{ss} = 1$ und $\tau_{sj} = 0$ für $j \in \overline{Z}$, $j \neq s$.
Sei $x \in L$ beliebig. Dann gilt

$$\sum_{s \in \overline{Z}} x_s^0 a^s = b = \sum_{j=1}^{n} x_j a^j = \sum_{j=1}^{n} x_j \left(\sum_{s \in \overline{Z}} \tau_{sj} a^s \right) = \sum_{s \in \overline{Z}} \left(\sum_{j=1}^{n} \tau_{sj} x_j \right) a^s$$

Da die a^s linear unabhängig sind, ist ein Koeffizientenvergleich möglich:

$$x_s^0 = \sum_{j=1}^{n} \tau_{sj} x_j = \sum_{j \notin \overline{Z}} \tau_{sj} x_j + x_s \text{ für } s \in \overline{Z}$$

Durch Umformung ergibt sich damit

$$x_s = x_s^0 - \sum_{j \notin \overline{Z}} \tau_{sj} x_j \text{ für } s \in \overline{Z}$$

Damit erhält man für die Zielfunktion z

$$\begin{aligned} z(x) &= \sum_{j=1}^{n} c_j x_j \\ &= \sum_{s\in\overline{Z}} c_s x_s + \sum_{j\notin\overline{Z}} c_j x_j \\ &= \sum_{s\in\overline{Z}} c_s x_s^0 - \sum_{j\notin\overline{Z}} \left(\sum_{s\in\overline{Z}} \tau_{sj} c_s - c_j \right) x_j \end{aligned}$$

Setze nun

$$d_j = \sum_{s\in\overline{Z}} \tau_{sj} c_s \text{ für} j \notin \overline{Z} \qquad \text{(II.2)}$$

Dann gilt

$$z(x) = z(x^0) - \sum_{j\notin\overline{Z}} (d_j - c_j) x_j \qquad \text{(II.3)}$$

Es können nun drei Fälle eintreten:

1. $d_j \leq c_j$ für alle $j \notin \overline{Z}$
   Dann ist $z(x) \geq z(x^0)$ für alle $x \in L$. x^0 *ist also optimale Lösung.*

2. Es existiert ein $i \notin \overline{Z}$ mit $d_i > c_i$ und $\tau_{si} \leq 0$ für alle $s \in \overline{Z}$.
   Sei $\delta > 0$ beliebig. Definiere $x^{(\delta)} = (x_1^{(\delta)}, \ldots, x_n^{(\delta)})^T$ durch

$$\begin{aligned} x_s^{(\delta)} &= x_s^0 - \delta\tau_{si} \text{ für } s \in \overline{Z} \\ x_i^{(\delta)} &= \delta \\ x_t^{(\delta)} &= 0 \text{ für } t \notin \overline{Z}, t \neq i \end{aligned}$$

   Dann ist $x^{(\delta)} \geq 0$. Außerdem ist $x^{(\delta)}$ eine Lösung von $Ax = b$, denn

$$\begin{aligned} Ax^{(\delta)} &= \sum_{s\in\overline{Z}} x_s^{(\delta)} a^s + \delta a^i \\ &= \sum_{s\in\overline{Z}} x_s^0 a^s - \delta \sum_{s\in\overline{Z}} \tau_{si} a^s + \delta a^i \\ &= b + \delta \underbrace{\left(a^i - \sum_{s\in\overline{Z}} \tau_{si} a^s \right)}_{=0} \text{ wegen (II.1)} \\ &= b \end{aligned}$$

   Also $x^{(\delta)} \in L$ für alle $\delta > 0$. Mit (II.3) folgt für die Zielfunktion z

$$z(x^{(\delta)}) = z(x^0) - \sum_{j\notin\overline{Z}} (d_j - c_j) x_j^{(\delta)}$$

$$= \quad z(x^0) - \underbrace{(d_i - c_i)}_{>0} \underbrace{\delta}_{>0}$$

Die Zielfunktion z ist also nicht nach unten beschränkt, d.h. *es existiert keine optimale Lösung.*

3. Es existiert ein $i \notin \overline{Z}$ und $k \in \overline{Z}$ mit $d_i > c_i$, $\tau_{ki} > 0$.
   Setze $\delta = \min\left\{x_l^0/\tau_{li} \mid l \in \overline{Z}, \tau_{li} > 0\right\}$ (nach Vorraussetzung existiert ein solches δ).
   Bilde $x^1 := x^{(\delta)}$ wie in Fall 2. $x^1 = (x_1^1, \ldots, x_n^1)^T$.
   Sei $r \in \overline{Z}$ mit $\delta = x_r^0/\tau_{ri}$. Für $l \in \overline{Z}$ mit $\tau_{li} > 0$ gilt also $x_r^0/\tau_{ri} \leq x_l^0/\tau_{li}$.
   Wir erhalten durch Umformung

$$0 \leq x_l^0 - \frac{x_r^0}{\tau_{ri}}\tau_{li} = x_l^1 \text{ (für } l = r \text{ ist } x_r^1 = 0)$$

   Für alle übrigen Indizes t ist trivialerweise $x_t^1 \geq 0$, also $x^1 \geq 0$.
   Man zeigt wie in Fall 2, dass x^1 Lösung von $Ax = b$ ist.
   Zeige nun: x^1 ist Ecke.
   Die zu x^1 gehörende Spaltenmenge ist in $\{a^s | s \in (\overline{Z} \setminus \{r\}) \cup \{i\}\}$ enthalten (Basiswechsel). Wäre diese Spaltenmenge linear abhängig, so hätten wir

$$\sum_{s \in \overline{Z} \setminus \{r\}} \mu_s a^s + \mu a^i = 0$$

   wobei nicht alle μ_s, μ gleich 0 sind.
   Dann ist also $\mu \neq 0$. Somit ergibt sich

$$a^i = \sum_{s \in \overline{Z} \setminus \{r\}} \left(-\frac{\mu_s}{\mu}\right) a^s$$

   Mit (II.1) und Koeffizientenvergleich folgt $\tau_{ri} = 0$, Widerspruch zur Wahl von r. Also ist die zu x^1 gehörende Spaltenmenge linear unabhängig. Nach Satz 2.18 ist also x^1 eine Ecke.

   Es müssen nun zwei Unterfälle betrachtet werden:

   (a) $\delta > 0$
   Nach (II.3) ist $z(x^1) = z(x^0) - \underbrace{(d_i - c_i)}_{>0} \underbrace{\delta}_{>0}$. Also $z(x^1) < z(x^0)$.

   (b) $\delta = 0$
   Nach (II.3) ist $z(x^1) = z(x^0)$.
   Dies passiert nur dann, wenn $x_r^0 = 0$ (d.h. x^0 ist entartete Ecke). Dann ist $x^1 = x^0$. Wir haben also nur die Basis von x^0 gewechselt.
   Dieser Fall ist vermeidbar durch geeignete Ordnung und Auswahl der Basen. Hierauf soll an dieser Stelle nicht näher eingegangen werden.

Wenn man dafür sorgt, dass Fall 3b nicht auftritt, so folgt aus der Endlichkeit der Eckenzahl (Korollar 2.20), dass der Algorithmus nach endlich vielen Schritten Fall 1 (optimale Lösung) oder Fall 2 (es existiert keine optimale Lösung) erreicht.

Teil (I) des Algorithmus
Wie findet man eine Ecke x^0 von L?
Zwei Fälle:

1. Ursprüngliches Problem ist ein Standard-Maximum-Programm

$$\begin{array}{l} Ax \leq b,\ x \geq 0 \\ z(x) = c^T x \text{ maximal} \end{array}$$

Angenommen $b \geq 0$. Wir führen nun Schlupfvariablen $z_1, \ldots, z_m$ ein, sodass

$$\begin{array}{l} \left(\begin{array}{c|c} A & E_m \end{array} \right) \left(\dfrac{x}{z} \right) = b,\ \left(\dfrac{x}{z} \right) \geq 0 \text{ wobei } z = \begin{pmatrix} z_1 \\ \vdots \\ z_m \end{pmatrix} \\ \left(\begin{array}{c|c} -c^T & 0^T \end{array} \right) \left(\dfrac{x}{z} \right) \text{ minimal} \end{array}$$

Dann ist $x^0 = \left(\dfrac{0}{b} \right)$ eine zulässige Lösung. Die zu x^0 gehörende Spaltenmenge besteht aus den Spalten von E_m, ist also linear unabhängig. Somit ist x^0 eine Ecke von L.

2. Programm ist bereits in der Form

$$(*) \quad \begin{array}{l} Ax = b,\ x \geq 0 \\ z(x) = c^T x \text{ minimal} \end{array}$$

Wir können $b \geq 0$ annehmen (ggf. entsprechende Zeilen von $Ax = b$ mit -1 multiplizieren).
Löse zunächst folgendes lineares Optimierungsproblem:

$$(**) \quad \begin{array}{l} \left(\begin{array}{c|c} A & E_m \end{array} \right) \left(\dfrac{x}{z} \right) = b,\ \left(\dfrac{x}{z} \right) \geq 0 \\ 0x_1 + \cdots + 0x_n + z_1 + \cdots + z_m \text{ minimal} \end{array}$$

Dabei ist offensichtlich $x^0 = \left(\dfrac{0}{b} \right)$ eine zulässige Lösung. Außerdem sind die Spalten der zu x^0 gehörenden Spaltenmenge linear unabhängig (entsprechen den Spalten von E_m). Nach Satz 2.18 ist x^0 somit eine Ecke von L.
Die Zielfunktion von $(**)$ ist auf der Menge der zulässigen Lösungen (durch

0) nach unten beschränkt. Nach Korollar 2.10 existiert also eine optimale Lösung, die in einer Ecke $\begin{pmatrix} x^* \\ z^* \end{pmatrix}$ von L liegt und durch den Simplex-Algorithmus bestimmbar ist.
Auch hier lassen sich nun wieder zwei Fälle unterscheiden:

a) $z^* \neq 0$
Der Minimalwert der Zielfunktion von $(**)$ ist also größer 0. In diesem Fall besitzt $(*)$ keine zulässige Lösung.
Denn angenommen $(*)$ besäße doch eine Lösung x. Dann wäre $\begin{pmatrix} x \\ 0 \end{pmatrix}$ eine Lösung von $(**)$ mit 0 als Minimalwert der Zielfunktion. Widerspruch!

b) $z^* = 0$
In diesem Fall ist $x^* \in L$, denn

$$b = \left(\begin{array}{c|c} A & E_m \end{array} \right) \begin{pmatrix} x^* \\ z^* \end{pmatrix} = \left(\begin{array}{c|c} A & E_m \end{array} \right) \begin{pmatrix} x^* \\ 0 \end{pmatrix} = Ax^*$$

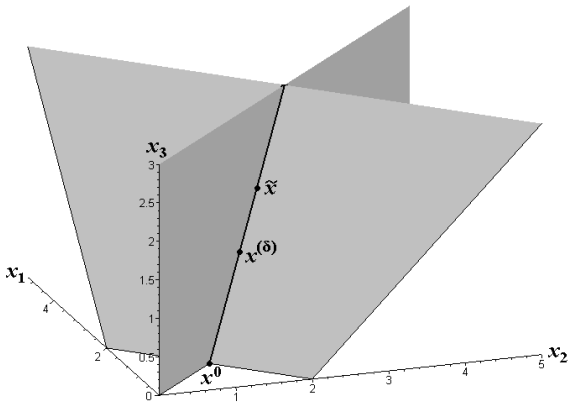
Außerdem ist x^* nach Satz 2.18 eine Ecke der Lösungsmenge von $(*)$, da $\begin{pmatrix} x^* \\ 0 \end{pmatrix}$ eine Ecke der Lösungsmenge von $(**)$ ist und die zugehörige linear unabhängige Spaltenmenge nur Spalten aus A enthält.

Geometrische Veranschaulichung von Teil (II) des Simplex-Algorithmus (Beispiele 2.5, 2.6)

Beispiel 2.5:
Die Menge der zulässigen Lösungen L sei gegeben durch

$$Ax = b,\ x \geq 0,\ \text{mit } A = \begin{pmatrix} 1 & 1 & -1 \\ 1 & -1 & 0 \end{pmatrix},\ x \geq 0 \text{ und } b = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

Nachfolgende Abbildung zeigt ausschnittsweise die beiden dadurch definierten Flächen, sowie den beim Schnitt entstehenden „Lösungsstrahl“.



Dabei ist

$$x^0 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix},\ \tilde{x} = \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} \in L$$

Die Gleichung für den Lösungsstrahl lautet

$$x^0 + \lambda(\tilde{x} - x^0) = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix},\ \lambda \geq 0$$

Spalten 1 und 2 der Matrix A bilden die Basis von x^0:

$$a^1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix},\ a^2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Wir stellen nun a^3 als Linearkombination von a^1 und a^2 dar und erhalten

$$a^3 = \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \tau_{13}a^1 + \tau_{23}a^2 \text{ mit } \tau_{13} = \tau_{23} = -\frac{1}{2} \qquad (\dagger)$$

x^0 und $\tilde{x}$ liegen auf dem Lösungsstrahl, d.h. $Ax^0 = b$ und $A\tilde{x} = b$. Damit ergibt sich $A(x^0 - \tilde{x}) = 0$. Dies lässt sich schreiben als:

$$a^1(x_1^0 - \tilde{x}_1) + a^2(x_2^0 - \tilde{x}_2) + a^3(x_3^0 - \tilde{x}_3) = 0$$

Ersetze a^3 durch $(\dagger)$:

$$a^1((x_1^0 - \tilde{x}_1) + \tau_{13}(x_3^0 - \tilde{x}_3)) + a^2((x_2^0 - \tilde{x}_2) + \tau_{23}(x_3^0 - \tilde{x}_3)) = 0$$

Da a^1 und a^2 linear unabhängig sind, muss gelten

$$\begin{aligned} x_1^0 - \tilde{x}_1 &= -\tau_{13}(x_3^0 - \tilde{x}_3) \\ x_2^0 - \tilde{x}_2 &= -\tau_{23}(x_3^0 - \tilde{x}_3) \end{aligned}$$

Wir setzen dies in den λ-Teil des Lösungsstrahls ein und erhalten

$$\lambda(\tilde{x} - x^0) = \lambda \begin{pmatrix} -\tau_{13}(x_3^0 - \tilde{x}_3) \\ -\tau_{23}(x_3^0 - \tilde{x}_3) \\ x_3^0 - \tilde{x}_3 \end{pmatrix} = \delta \begin{pmatrix} -\tau_{13} \\ -\tau_{23} \\ 1 \end{pmatrix}$$

Wir erhalten damit $x^{(\delta)}$ mit

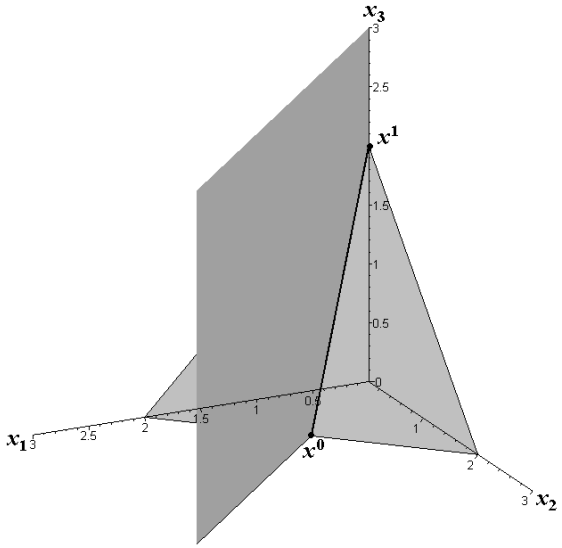
$$\begin{aligned} x_1^{(\delta)} &= x_1^0 - \delta\tau_{13} \\ x_2^{(\delta)} &= x_2^0 - \delta\tau_{23} \\ x_3^{(\delta)} &= \delta \end{aligned}$$

Die zweite Bedingung in Fall 2 ($d_3 > c_3$) sorgt dafür, dass z entlang des Strahls immer kleiner wird.

BEISPIEL 2.6:
Die Menge der zulässigen Lösungen L sei gegeben durch

$$Ax = b \text{ mit } A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \end{pmatrix} \text{ und } b = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

Nachfolgende Abbildung zeigt wiederum ausschnittsweise die beiden dadurch definierten Flächen, sowie die beim Schnitt entstehende „Lösungsstrecke“.



Dabei sind die beiden Ecken dieser Lösungsmenge

$$x^0 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \; x^1 = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}$$

In diesem Fall ergibt sich nun für τ_{13} und τ_{23} jeweils $\frac{1}{2}$. Für die Strecke zwischen x^0 und x^1 ergibt sich die Parametergleichung

$$x^0 + \delta \begin{pmatrix} -\tau_{13} \\ -\tau_{23} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \delta \begin{pmatrix} -1/2 \\ -1/2 \\ 1 \end{pmatrix}, \; 0 \leq \delta \leq \min \left\{ \frac{x_1^0}{\tau_{13}}, \frac{x_2^0}{\tau_{23}} \right\} = 2$$

Jetzt hängt es nur davon ab, ob $z(x^1) > z(x^0)$, d.h. ob $d_3 > c_3$. Wenn ja, so 3. Fall, wenn nein, so Optimum bei x^0 (1. Fall).

BEISPIEL 2.7:
Standard-Maximum-Programm

$$\begin{array}{rcrcrcl} 2x_1 & - & x_2 & + & 5x_3 & \leq & 6 \\ x_1 & + & x_2 & - & x_3 & \leq & 2 \end{array}, \; x_1, x_2, x_3 \geq 0$$
$$\tilde{z}(x) = 3x_1 - x_2 - x_3 \text{ maximal}$$

Durch Einführung von Schlupfvariablen erhalten wir folgendes kanonische Minimum-Programm:

$$\begin{array}{rcrcrcrcrcl} 2x_1 & - & x_2 & + & 5x_3 & + & x_4 & & & = & 6 \\ x_1 & + & x_2 & - & x_3 & & & + & x_5 & = & 2 \end{array} \quad , \; x_1, x_2, x_3, x_4, x_5 \geq 0$$
$$z(x) = -3x_1 + x_2 + x_3 \text{ minimal}$$

Also:

$$A = \begin{pmatrix} 2 & -1 & 5 & 1 & 0 \\ 1 & 1 & -1 & 0 & 1 \end{pmatrix}, \; c = \begin{pmatrix} -3 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \; b = \begin{pmatrix} 6 \\ 2 \end{pmatrix}$$

1. Schritt
$x^0 = (0, 0, 0, 6, 2)^T$ mit $z(x^0) = 0$, $\overline{Z} = Z = \{4, 5\}$
Basis von $x^0 = \{a^4 = \binom{1}{0}, a^5 = \binom{0}{1}\}$
Darstellung von a^1, a^2, a^3 als Linearkombinationen von a^4, a^5:

$$\begin{array}{rcrcr} a^1 & = & 2a^4 & + & a^5 \\ a^2 & = & -a^4 & + & a^5 \\ a^3 & = & 5a^4 & - & a^5 \end{array}$$

Tableau der τ_{sj}, $s \in \overline{Z}, j = 1, \ldots, 5$:

	1	2	3	4	5
4	2	−1	5	1	0
5	1	1	−1	0	1

Berechnung der $d_j = \sum_{s \in \overline{Z}} \tau_{sj} c_s$ für $j \notin \overline{Z}$:

$$\begin{array}{rcrcrcl} d_1 & = & \tau_{41} c_4 & + & \tau_{51} c_5 & = & 0 \\ d_2 & = & \tau_{42} c_4 & + & \tau_{52} c_5 & = & 0 \\ d_3 & = & \tau_{43} c_4 & + & \tau_{53} c_5 & = & 0 \end{array}$$

$0 = d_1 > c_1 = -3$, $\tau_{41}, \tau_{51} > 0$. Also ist Fall 3 eingetreten:

$$\delta = \min\left\{\frac{x_4^0}{\tau_{41}}, \frac{x_5^0}{\tau_{51}}\right\} = \min\left\{\frac{6}{2}, \frac{2}{1}\right\}$$

Also ist $\delta = 2$ mit $r = 5$ und $i = 1$.

Berechnung von x^1:

$$\begin{array}{rcll} x_s^1 & = & x_s^0 - \delta\tau_{si}, \; s \in \overline{Z} & \Rightarrow x_4^1 = 6 - 2 \cdot 2 = 2, \; x_5^1 = 0 \\ x_i^1 & = & \delta & \Rightarrow x_1^1 = 2 \\ x_t^1 & = & 0 & \Rightarrow x_2^1 = 0, \; x_3^1 = 0 \end{array}$$

Also ist $x^1 = (2,0,0,2,0)^T$ mit $z(x^1) = -6$.

2. Schritt
$x^1 = (2,0,0,2,0)^T$ mit $z(x^1) = -6$, $\overline{Z} = Z = \{1,4\}$
Basis von $x^1 = \{a^1 = \binom{2}{1}, a^4 = \binom{1}{0}\}$
Darstellung von a^2, a^3, a^5 als Linearkombinationen von a^1, a^4:

$$\begin{array}{lcrcr} a^2 & = & a^1 & - & 3a^4 \\ a^3 & = & -a^1 & + & 7a^4 \\ a^5 & = & a^1 & - & 2a^4 \end{array}$$

Tableau der τ_{sj}, $s \in \overline{Z}, j = 1,\ldots,5$:

	1	2	3	4	5
1	1	1	−1	0	1
4	0	−3	7	1	−2

Berechnung der $d_j = \sum_{s\in\overline{Z}} \tau_{sj} c_s$ für $j \notin \overline{Z}$:

$$\begin{array}{lcrcrcr} d_2 & = & \tau_{12}c_1 & + & \tau_{42}c_4 & = & -3 \\ d_3 & = & \tau_{13}c_1 & + & \tau_{43}c_4 & = & 3 \\ d_5 & = & \tau_{15}c_1 & + & \tau_{45}c_4 & = & -3 \end{array}$$

$3 = d_3 > c_3 = 1$, $\tau_{43} = 7 > 0$ ($\tau_{13} = -1$). Also ist Fall 3 eingetreten: $\delta = \frac{x_4^1}{\tau_{43}} = \frac{2}{7}$ mit $r = 4$ und $i = 3$.

Berechnung von x^2:

$$\begin{array}{lcll} x_s^2 & = & x_s^1 - \delta\tau_{si},\ s \in \overline{Z} & \Rightarrow x_1^2 = 2 - \frac{2}{7}\cdot(-1) = \frac{16}{7},\ x_4^2 = 0 \\ x_i^2 & = & \delta & \Rightarrow x_3^2 = \frac{2}{7} \\ x_t^2 & = & 0 & \Rightarrow x_2^2 = 0,\ x_5^2 = 0 \end{array}$$

Also ist $x^2 = \left(\frac{16}{7}, 0, \frac{2}{7}, 0, 0\right)^T$ mit $z(x^2) = -\frac{46}{7}$.

3. Schritt
$x^2 = \left(\frac{16}{7}, 0, \frac{2}{7}, 0, 0\right)^T$ mit $z(x^2) = -\frac{46}{7}$, $\overline{Z} = Z = \{1,3\}$
Basis von $x^2 = \{a^1 = \binom{2}{1}, a^3 = \binom{5}{-1}\}$
Darstellung von a^2, a^4, a^5 als Linearkombinationen von a^1, a^3:

$$\begin{array}{lcrcr} a^2 & = & \frac{4}{7}a^1 & - & \frac{3}{7}a^3 \\ a^4 & = & \frac{1}{7}a^1 & + & \frac{1}{7}a^3 \\ a^5 & = & \frac{5}{7}a^1 & - & \frac{2}{7}a^3 \end{array}$$

Tableau der τ_{sj}, $s \in \overline{Z}, j = 1,\ldots,5$:

	1	2	3	4	5
1	1	$\frac{4}{7}$	0	$\frac{1}{7}$	$\frac{5}{7}$
3	0	$-\frac{3}{7}$	1	$\frac{1}{7}$	$-\frac{2}{7}$

Berechnung der $d_j = \sum_{s \in \overline{Z}} \tau_{sj} c_s$ für $j \notin \overline{Z}$:

$$\begin{array}{lllllll} d_2 & = & \tau_{12}c_1 & + & \tau_{32}c_3 & = & -\frac{15}{7} \\ d_4 & = & \tau_{14}c_1 & + & \tau_{34}c_3 & = & -\frac{2}{7} \\ d_5 & = & \tau_{15}c_1 & + & \tau_{35}c_3 & = & -\frac{17}{7} \end{array}$$

$-\frac{15}{7} = d_2 < c_2 = 1$, $-\frac{2}{7} = d_4 < c_4 = 0$, $-\frac{17}{7} = d_5 < c_5 = 0$. Also ist Fall 1 eingetreten: x^2 ist optimale Lösung des kanonischen Minimum-Programms.

Für das ursprüngliche Standard-Maximum-Programm ergibt sich damit:
$\tilde{x} = \left(\frac{16}{7}, 0, \frac{2}{7}\right)^T$ ist optimale Lösung mit einem maximalen Wert von $\tilde{z} = \frac{46}{7}$ für die Zielfunktion.

2.23 Bemerkung

1. Der Simplex-Algorithmus, entwickelt 1947 von Dantzig, funktioniert über $\mathbb{R}$ und $\mathbb{Q}$.

2. Komplexität des Simplex-Algorithmus
   Worst-Case-Komplexität ist $O(2^n)$ (Klee, Minty, 1972).
   Durchschnittliche Komplexität ist polynomial (Borgwardt, 1982). Unter einer plausiblen Verteilungsannahme führt der Algorithmus $O(n^3 m^{\frac{1}{n-1}})$ viele Basiswechsel durch.
   Fazit: In der Praxis ist der Simplex-Algorithmus sehr tauglich.

3. Polynomiale Algorithmen zur Lösung linearer Optimierungsprobleme (vgl. [Sch99], Kapitel 11.4, 11.5, 13 und 15.1):

   - Khachian, 1979: Ellipsoid-Methode[3] (in der Regel schlechter als der Simplex-Algorithmus)
   - Karmarkar, 1984

[3] Ursprünglich wurde die Ellipsoid-Methode von Shor, Yudin, Nemirowskii für die Lösung nicht-linearer Optimierungsprobleme entwickelt.

Kapitel 3

Ganzzahlige lineare Optimierung

3.1 Definition (Normalformen ganzzahliger Maximum-Probleme)

a) Gegeben $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, sowie Zielfunktion $z(x) = c^T x$, $x \in \mathbb{Q}^n$. Bestimme $\max\{z(x) \mid x \in \mathbb{Z}^n,\ Ax \leq b\}$, falls ein solches Maximum existiert. Bestimme gegebenenfalls außerdem $x_0 \in \mathbb{Z}^n$ mit $Ax_0 \leq b$, $z(x_0) \geq z(x)$ für alle $x \in \mathbb{Z}^n$.

b) wie a), aber mit $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$

c) Gegeben $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, sowie Zielfunktion $z(x) = c^T x$, $x \in \mathbb{Q}^n$. Bestimme $\max\{z(x) \mid x \in \mathbb{Z}^n,\ Ax = b\}$, falls ein solches Maximum existiert. Bestimme gegebenenfalls außerdem $x_0 \in \mathbb{Z}^n$ mit $Ax_0 = b$, $z(x_0) \geq z(x)$ für alle $x \in \mathbb{Z}^n$.

d) wie c), aber mit $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$

e) Gegeben $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$, sowie Zielfunktion $z(x) = c^T x$, $x \in \mathbb{Q}^n$. Bestimme $\max\{z(x) \mid x \in \mathbb{Z}^n,\ Ax \leq b,\ x \geq 0\}$, falls ein solches Maximum existiert. Bestimme gegebenenfalls außerdem $x_0 \in \mathbb{Z}^n$ mit $Ax_0 \leq b$, $x_0 \geq 0$, $z(x_0) \geq z(x)$ für alle $x \in \mathbb{Z}^n$.

3.2 Bemerkung
Die Aufgaben a) bis e) aus Definition 3.1 sind äquivalent.

Beweis.

1. Zeige: a) $\Leftrightarrow$ b), c) $\Leftrightarrow$ d)
   Multipliziere mit positivem Hauptnenner der in A, b, c vorkommenden Brüche.

2. Zeige: Lösbarkeit von c) impliziert Lösbarkeit von b)

Betrachte

$$\max\left\{ \left(\begin{array}{c|c|c} c^T & -c^T & 0^T \end{array}\right) \begin{pmatrix} x' \\ x'' \\ \tilde{x} \end{pmatrix} \mid \left(\begin{array}{c|c|c} A & -A & E_m \end{array}\right) \begin{pmatrix} x' \\ x'' \\ \tilde{x} \end{pmatrix} = b \right\}$$

wobei $x', x'' \in \mathbb{Z}^n$, $\tilde{x} \in \mathbb{Z}^m$, $x', x'', \tilde{x} \geq 0$. Ist $x \in \mathbb{Z}^n$ Lösung von $Ax \leq b$, so schreibe $x = x' - x''$ mit $x', x'' \in \mathbb{Z}^n$, $x', x'' \geq 0$. Damit ergibt sich dann $Ax' - Ax'' = Ax \leq b$.
Führe Schlupfvariablen ein, sodass $Ax' - Ax'' + \tilde{x} = b$ für ein $\tilde{x} \geq 0$, $\tilde{x} \in \mathbb{Z}^m$.
$(x'^T, x''^T, \tilde{x}^T)^T$ erfüllt die Bedingungen der oben beschriebenen Menge, von der das Maximum gesucht wird und ist somit also Lösung eines c)-Problems.
Liegt umgekehrt $(x'^T, x''^T, \tilde{x}^T)^T$ in der oben angegebenen Menge, so erfüllt $x = x' - x''$ die Bedingung $x \in \mathbb{Z}^n$ und $Ax \leq b$ (da $\tilde{x} \geq 0$). Der Wert der Zielfunktion bleibt gleich.

3. Zeige: Lösbarkeit von a) impliziert Lösbarkeit von c)
   $Ax = b$, $x \geq 0$ ist äquivalent zu $-x \leq 0$, $Ax \leq b$, $(-A)x \leq -b$. Damit lässt sich folgendes a)-Problem konstruieren:

$$\begin{pmatrix} -E_n \\ A \\ -A \end{pmatrix} x \leq \begin{pmatrix} 0 \\ b \\ -b \end{pmatrix}$$

4. Zeige: b) $\Leftrightarrow$ e)
   $x = x' - x''$, $x', x'' \in \mathbb{Z}^n$, $x', x'' \geq 0$
   $Ax \leq b \;\Leftrightarrow\; \left(\begin{array}{c|c} A & -A \end{array}\right) \begin{pmatrix} x' \\ x'' \end{pmatrix} \leq b$, $c^T x = \left(\begin{array}{c|c} c^T & -c^T \end{array}\right) \begin{pmatrix} x' \\ x'' \end{pmatrix}$ ∎

3.3 Definition (Normalformen ganzzahliger Minimum-Probleme)
Analog den Maximum-Problemen.

Frage: Kann man die Lösung des linearen Optimierungsproblems $\max\{z(x) \mid x \in \mathbb{Q}^n, Ax \leq b,\ x \geq 0\}$ zur Bestimmung der Lösung des ganzzahligen linearen Optimierungsproblems $\max\{z(x) \mid x \in \mathbb{Z}^n,\ Ax \leq b,\ x \geq 0\}$ verwenden?

Beispiel 3.1:
Das ganzzahlige lineare Optimierungsproblem lautet:

$$\begin{pmatrix} -3 & 10 \\ 0 & -10 \\ 10 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 9 \\ -9 \\ 45 \end{pmatrix},\ x_i \in \mathbb{Z},\ x_i \geq 0 \text{ für } i = 1, 2$$

$$z(x) = -10x_1 + 31x_2 \text{ maximal}$$

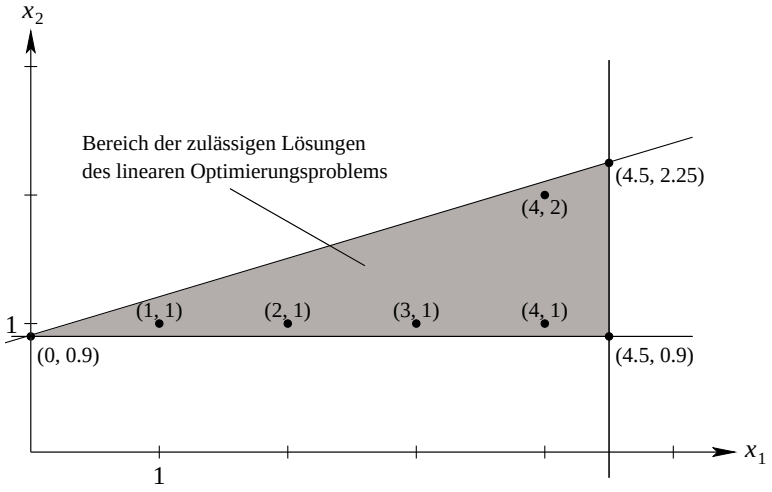


Abbildung 3.1: Zulässige Lösungen des in Beispiel 3.1 angegebenen Optimierungsproblems

Abbildung 3.1 zeigt den Bereich der zulässigen Lösungen des zugehörigen linearen Optimierungsproblems mit $x_i \in \mathbb{Q}$. Als Maximum ergibt sich für das LP $27,9$ bei $x = \binom{0}{0,9}$. Der nächste ganzzahlige Punkt $\binom{0}{1}$ liegt nicht im Bereich der zulässigen Lösungen.
Als Maximum für das beschriebene ganzahlig lineare Optimierungsproblem ergibt sich 22 bei $x = \binom{4}{2}$. Dies ist der von $\binom{0}{0,9}$ am weitesten entfernteste ganzzahlige Punkt in L.

Dennoch kann die Betrachtung des zu einem ganzzahligen linearen Opimierungsproblem gehörende lineare Optimierungsproblem helfen. Wir bezeichnen im folgenden häufig ganzzahlige lineare Optimierungsprobleme mit *ILP* (*Integer Linear Programming*) und lineare Optimierungsprobleme (über $\mathbb{Q}$) mit *LP* (*Linear Programming*).

3.4 Definition
Gegeben sei das ILP $\max\{c^T x \mid Ax \leq b,\ x \in \mathbb{Z}^n,\ x \geq 0\}$, wobei $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$.
Werden die Nebenbedingungen gelockert (z.B. $x_i \in \mathbb{Q}$ statt $x_i \in \mathbb{Z}$) oder weggelassen (z.B. eine Ungleichung), so spricht man von einer *Relaxation* des ILP (damit ist die neue Lösungsmenge größer oder gleich der alten Lösungsmenge). Falls man beim ILP die Bedingung $x \in \mathbb{Z}^n$ durch $x \in \mathbb{Q}^n$ ersetzt, so spricht man von der *LP-Relaxation* des ILP.

3.5 Bemerkung
a) Für Maximum-Probleme gilt:
Optimaler Wert des ILP $\leq$ Optimaler Wert der LP-Relaxation

Für Minimum-Probleme gilt analog:
Optimaler Wert des ILP $\geq$ Optimaler Wert der LP-Relaxation

b) Wenn die optimale Lösung der LP-Relaxation ganzzahlig ist, dann ist dies auch eine optimale Lösung des ILP.

c) Wenn die LP-Relaxation keine zulässigen Lösungen besitzt, so gilt dies auch für das zugehörige ILP.

d) Hat die Zielfunktion des ILP ganzzahlige Koeffizienten (wie in Definition 3.4 gefordert), so gilt: Ist x optimale Lösung der LP-Relaxation, so ist der optimale Wert des ILP $\leq \lfloor z(x) \rfloor$ (falls ein solcher optimaler Wert existiert).

e) Ist y_0 zulässige Lösung des zur LP-Relaxation gehörenden dualen Programms ($A^T y \geq c$, $y \geq 0$, $z^*(y) = b^T y$ minimal), dann ist der optimale Wert des ILP $\leq b^T y_0$ (falls ein solcher optimaler Wert existiert). Dies folgt aus Satz 2.3.

3.1 Branch-and-Bound

Branch-and-Bound (B&B) ist ein allgemeines Aufzählungsverfahren, das 1965 von Dakin das erste Mal im ILP-Kontext verwendet wurde.
Grundprinzip: Das Gesamtproblem wird (ggf. wiederholt) in Teilprobleme zerlegt (branching), wobei sich eine Baumstruktur ergibt. Anschließend werden Schranken (bounds) für die einzelnen Teilprobleme berechnet. Aufgrund dieser Bounds müssen dann anschließend manche der Teilprobleme weiterverfolgt werden, andere nicht.

3.6 Allgemeines Vorgehen bei Branch-and-Bound

(1) Löse die LP-Relaxation des ILP. Ist die Lösung ganzzahlig, endet der Algorithmus. Andernfalls verwendet der Algorithmus die optimale Lösung der LP-Relaxation als *Bound* und erzeugt zwei neue Teilprobleme (*Branching*) bei einer der Variablen, die bei der optimalen Lösung der LP-Relaxation nicht ganzzahlig waren (z.B. werden aus $x_i = 4,5$ Teilprobleme mit $x_i \leq 4$ und $x_i \geq 5$ erzeugt)

(2) Ein Teilproblem ist *nicht aktiv*, falls einer der folgenden Fälle eintritt:

   (a) Teilproblem wurde bereits zum Branching verwendet.

   (b) Teilproblem hat ganzzahlige optimale Lösung.

   (c) Teilproblem hat keine zulässigen Lösungen.

   (d) Teilproblem hat Schranke (Bound) für die optimale Lösung, die kleiner als eine schon gefundene ganzzahlige Lösung ist.

(3) Wähle eines der aktiven Teilprobleme und verfahre wie in 1. Falls keine aktiven Teilprobleme mehr existieren, so ist die optimale Lösung des ILP

die größte ganzzahlige Lösung eines der Teilprobleme. Falls keine solche existiert, existiert auch keine optimale Lösung.

Dass der Branch-and-Bound-Algorithmus terminiert, werden wir in 3.10 zeigen. Dort wird insbesondere gezeigt, wie der kritische Fall, dass die Zielfunktion auf der Menge der zulässigen Lösungen des ILP unbeschränkt ist, erkannt wird.

BEISPIEL 3.2:
Vier Gegenstände sollen zu einem Käufer transportiert werden. Das Gewicht der Gegenstände ist 5t, 7t, 4t, 3t. Ihr Verkaufswert ist 8000 Euro, 11000 Euro, 6000 Euro, 4000 Euro. Es steht allerdings nur ein Lkw mit einer Nutzlast von 14t zur Verfügung. Welche Gegenstände sollen ausgewählt werden, sodass der Verkaufswert maximal wird?

Problem: Maximiere $z(x) = 8x_1 + 11x_2 + 6x_3 + 4x_4$ unter der Bedingung $5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$, wobei $x_i \in \{0, 1\}$, d.h. $0 \leq x_i \leq 1,\ x_i \in \mathbb{Z}$.

Klar: Es gibt nur 2^4 Lösungsmöglichkeiten, d.h. eine vollständige Durchsuchung des Lösungsraumes wäre hier ohne Weiteres möglich.

Abbildung 3.2 zeigt graphisch die Vorgehensweise des Algorithmus.

I: Löse die LP-Relaxation des oben angegebenen ILP (z.B. mit Hilfe des Simplex-Algorithmus). Es ergibt sich ein optimaler Wert $z = 22$ für $x_1 = 1$, $x_2 = 1$, $x_3 = \frac{1}{2}$, $x_4 = 0$. Dabei handelt es sich um keine ganzzahlige Lösung. Allerdings wissen wir nun, dass der maximale Wert des ILP ≤ 22 ist.

II: Durch das Setzen von x_3 auf 0 ergibt sich aus dem ursprünglichen ILP folgendes Teilproblem:

$$\text{Maximiere } 8x_1 + 11x_2 + 4x_4 \text{ bezüglich } 5x_1 + 7x_2 + 3x_4 \leq 14$$
$$0 \leq x_1, x_2, x_4 \leq 1,\ x_i \in \mathbb{Z}$$

Der optimale Wert der zugehörigen LP-Relaxation ist $21\frac{2}{3}$ für $x_1 = 1$, $x_2 = 1$, $x_4 = \frac{2}{3}$.

III: Durch das Setzen von x_3 auf 1 ergibt sich aus dem ursprünglichen ILP folgendes Teilproblem:

$$\text{Maximiere } 8x_1 + 11x_2 + 4x_4 + 6 \text{ bezüglich } 5x_1 + 7x_2 + 3x_4 \leq 10$$
$$0 \leq x_1, x_2, x_4 \leq 1,\ x_i \in \mathbb{Z}$$

Der optimale Wert der zugehörigen LP-Relaxation ist $21\frac{6}{7}$ für $x_1 = 1$, $x_2 = \frac{5}{7}$, $x_4 = 0$. Wir sehen nun, dass sowohl in Fall II als auch in III der optimale Wert kleiner als 22 ist. Wir können nun also sagen, dass der optimale Wert des ursprünglichen ILP ≤ 21 sein muss.

Die weiteren in Abbildung 3.2 dargestellten Schritte laufen analog ab und liefern am Ende eine optimale Lösung $z = 21$ für $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$ (vgl. VI).

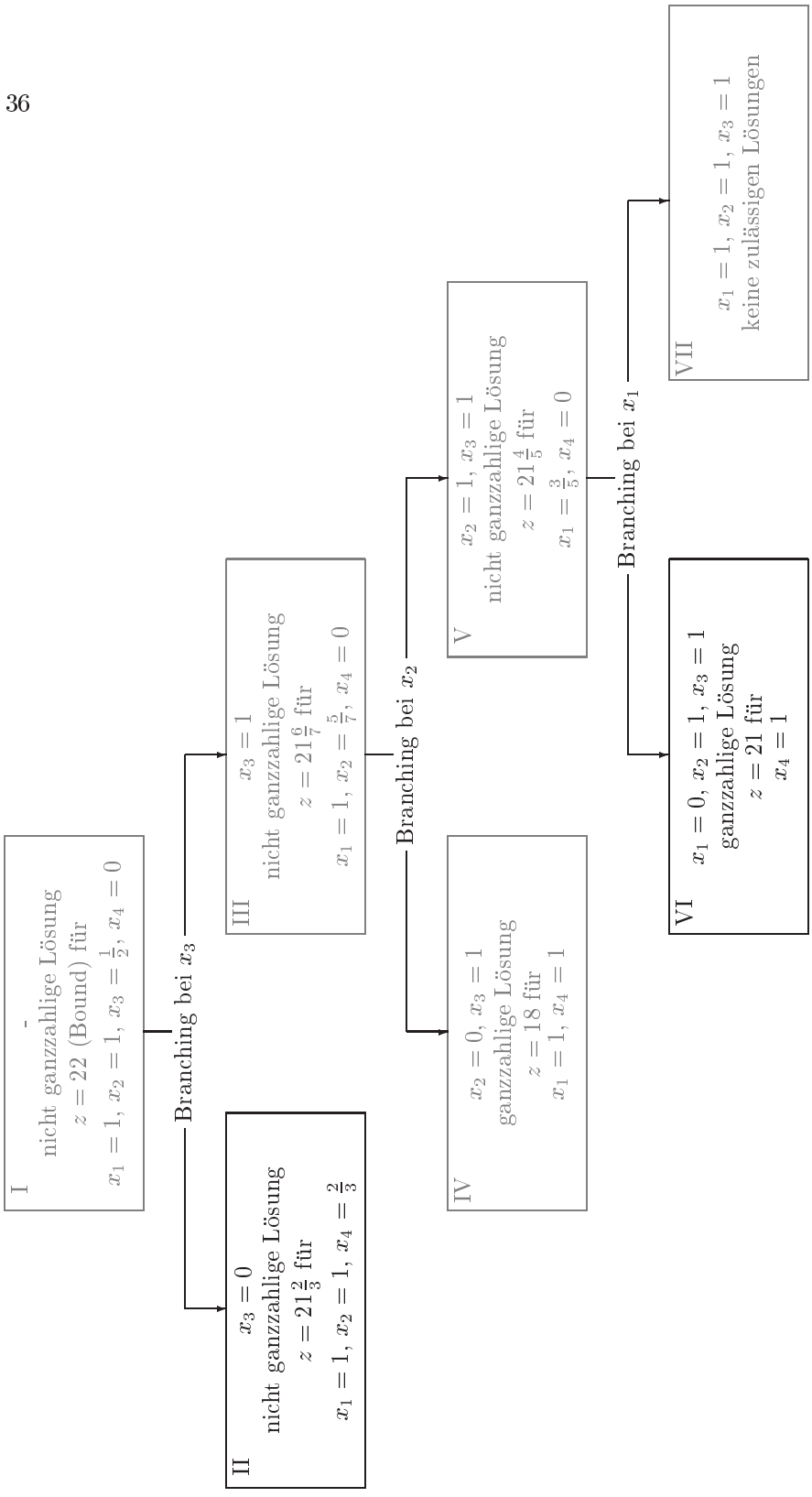


Abbildung 3.2: Ablauf des Branch-and-Bound-Algorithmus anhand von Beispiel 3.2. Die grauen Boxen stellen dabei solche Teilprobleme dar, die entweder bereits weiterverfolgt wurden oder die aufgrund ihres Bound-Wertes nicht weiter verfolgt werden müssen. Box Nummer VI enthält eine optimale ganzzahlige Lösung für das ILP.

3.7 Modifikationen und Heuristiken

1. Zur Verringerung des Aufwandes bei der Berechnung der optimalen Lösung der LP-Relaxation kann (insbesondere zu Beginn des Algorithmus) auch ein zulässiger Wert des dualen LP-Problems als Schranke verwendet werden (oder sogar $+\infty$ (bzw. $-\infty$)). Dann hat man allerdings keinen Hinweis, bei welcher Variablen sinnvollerweise verzweigt werden sollte.

2. Bei der Auswahl der aktiven Teilprobleme wird i.a. entweder Depth-First-Search oder Best-First-Choice (d.h. es wird das Teilproblem mit maximalem Bound ausgewählt) verwendet.
   Depth-First-Search führt dabei zum einen schnell zu ganzzahligen Lösungen, zum anderen existieren i.a. nur wenige aktive Teilprobleme. Best-First-Choice führt dagegen häufig schnell zu einer optimalen ganzzahligen Lösung.

3. Problemspezifisches Wissen kann häufig vorteilhaft eingesetzt werden (z.B. Branching bei „wichtigen“ Variablen).

3.2 ILP und LP-Relaxation; Terminierung von Branch-and-Bound

Zur Beantwortung der Frage, ob der Branch-and-Bound-Algorithmus überhaupt terminiert, werden einige Zusammenhänge zwischen ILP und der zugehörigen LP-Relaxation benötigt.

3.8 Satz
Sei das ILP gegeben durch $\max\{c^T x \mid Ax \leq b,\ x \geq 0,\ x \in \mathbb{Z}^n\}$, wobei $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$ und die zugehörige LP-Relaxation durch $\max\{c^T x \mid Ax \leq b,\ x \geq 0,\ x \in \mathbb{Q}^n\}$.
Es gilt dann:
Ist $z(x) = c^T x$ auf der Menge L der zulässigen Lösungen der LP-Relaxation nach oben unbeschränkt, so ist entweder

$$L \cap \mathbb{Z}^n = \emptyset \text{ (d.h. ILP hat keine zulässigen Lösungen)}$$
$$\text{oder}$$
$$z(x) \text{ ist auch auf } L \cap \mathbb{Z}^n \text{ unbeschränkt.}$$

Beweis.
Angenommen $L \cap \mathbb{Z}^n \neq \emptyset$, $x_0 \in L \cap \mathbb{Z}^n$.
Da $z(x)$ auf L unbeschränkt ist, folgt mit Bemerkung 2.11, dass das duale LP-Programm $A^T y \geq c$, $y \geq 0$ ($z^*(y) = b^T y$ minimal) keine zulässigen Lösungen $y \in \mathbb{Q}^n$ besitzt. Nach Satz 2.7 ist dann $-Au \geq 0$, $u \geq 0$, $-c^T u < 0$ lösbar mit $u \in \mathbb{Q}^n$.
Durch Umformung der Ungleichungen und Multiplikation mit dem (positiven) Hauptnenner der Einträge von u folgt:
Es existiert ein $r \in \mathbb{Z}^n$ mit $Ar \leq 0$, $r \geq 0$, $c^T r > 0$.

Mit r erfüllen auch alle kr mit $k \in \mathbb{N}$ diese Bedingungen. Damit erhalten wir folgende Aussage:

$$kr + x_0 \in \mathbb{Z}^n \text{ mit } kr + x_0 \geq 0,\ A(kr + x_0) = k\underbrace{Ar}_{\leq 0} + \underbrace{Ax_0}_{\leq b} \leq b$$

D.h. $kr + x_0 \in L \cap \mathbb{Z}^n$. Damit ist z auf der Menge der zulässigen Lösungen des ILP unbeschränkt, denn $z(kr + x_0) = k\underbrace{c^T r}_{>0} + c^T x_0 \to \infty$ für $k \to \infty$. ∎

Bezeichnungen
Ist $\alpha = \frac{p}{q} \in \mathbb{Q}$, so sei $l(\alpha)$ („size“) definiert durch

$$l(\alpha) = 1 + \underbrace{\lceil \log_2(|p| + 1) \rceil}_{\text{binäre Länge von } |p|} + \lceil \log_2(|q| + 1) \rceil$$

Ist $c = (\alpha_1, \ldots, \alpha_n)^T \in \mathbb{Q}^n$, so ist $l(c)$ definiert durch

$$l(c) = n + l(\alpha_1) + \ldots + l(\alpha_n)$$

Ist $A = (\alpha_{ij}) \in \mathbb{Q}^{m \times n}$, so ist $l(A)$ definiert durch

$$l(A) = mn + \sum_{i,j} l(\alpha_{ij})$$

3.9 Satz
Sei ILP und zugehörige LP-Relaxation gegeben wie in Satz 3.8.

a) Ist $L \cap \mathbb{Z}^n \neq \emptyset$, so existiert $u \in L \cap \mathbb{Z}^n$ mit $l(u) \leq 6n^3(1 + l(A) + l(b)) =: T$.

b) Ist $L \cap \mathbb{Z}^n \neq \emptyset$ und ist $z(x)$ auf $L \cap \mathbb{Z}^n$ beschränkt, so existiert u wie in a) mit $z(u) = \max\{z(x) \mid x \in L \cap \mathbb{Z}^n\}$.

(vgl. [Sch99], Corollar 17.1 b), c))

3.10 Terminierung von Branch-and-Bound
ILP und zugehörige LP-Relaxation seien gegeben wie in Satz 3.8. Ist die Lösungsmenge L der LP-Relaxation beschränkt, so enthält L nur endlich viele ganzzahlige Vektoren. Verzweigt man bei Branch-and-Bound stets bei nicht-ganzzahligen x_i, so ist klar, dass der Algorithmus nach endlicher Zeit terminiert.

Was ist, wenn man nicht weiß, ob $L = \{x \mid Ax \leq b,\ x \geq 0,\ x \in \mathbb{Q}^n\}$ beschränkt ist?
Setze $L_0 = L \cap \{x = (x_1, \ldots, x_n)^T \in \mathbb{Q}^n \mid x_i \geq 0,\ x_i \leq 2^T\}$, wobei T wie in Satz 3.9 definiert ist.
Bestimme nun mit Branch-and-Bound $\max\{z(x) \mid x \in L_0, x \in \mathbb{Z}^n\}$ $(*)$, falls ein solches Maximum existiert. Der Algorithmus terminiert, da L_0 beschränkt ist. Unterscheide zwei Fälle:

1. Es existiert keine Lösung: Nach Satz 3.9a) hat dann das ursprüngliche ILP auch keine Lösung.

2. $(*)$ besitzt eine optimale Lösung x^*: Überprüfe mit dem Simplex-Algorithmus, ob die LP-Relaxation des ursprünglichen ILP ein endliches Optimum besitzt, oder ob $z(x)$ auf L unbeschränkt ist:

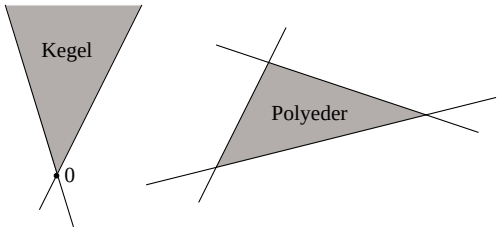
   (a) LP-Relaxation besitzt ein endliches Optimum: Nach Satz 3.9b) ist x^* eine optimale Lösung des ILP.

   (b) $z(x)$ ist auf L unbeschränkt: Wegen $x^* \in L \cap \mathbb{Z}^n \neq \emptyset$ ist nach Satz 3.8 $z(x)$ auch auf $L \cap \mathbb{Z}^n$ unbeschränkt.

3.11 Bemerkung
Es gelten dieselben Bezeichnungen wie in Satz 3.8.
Ist L beschränkt, so terminiert der Branch-and-Bound-Algorithmus. Die Laufzeit ist jedoch im Worst Case exponentiell (vgl. [Sch99], Kapitel 24.1).

3.12 Bemerkung
Satz 3.9 beruht auf einigen wichtigen Sätzen über rationale Polyeder und LP-/ILP-Probleme.

1. Ein (konvexes) *Polyeder* P in $\mathbb{R}^n$ ist von der Form $\{x \in \mathbb{R}^n \mid Ax \leq b\}$ für eine Matrix A und einen Vektor b. Ist $b = 0$ heißt P *konvexer Kegel*.
   Ein *Polytop* ist ein beschränktes Polyeder.



2. **Satz (Minkowski, Weyl, Motzkin)**
   Jeder Polyeder lässt sich schreiben in der Form

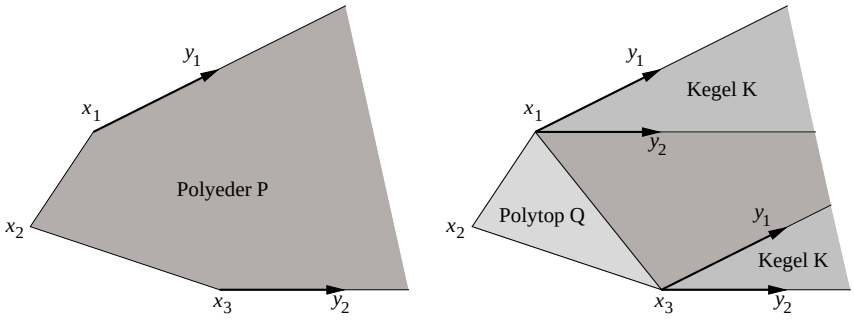
$$P = Q + K = \{q + k \mid q \in Q, k \in K\}$$

   wobei Q ein Polytop und K ein konvexer Kegel ist.
   Es existieren endlich viele Punkte $x_1, \ldots, x_r$, so dass

$$Q = \left\{ \sum_{i=1}^{r} \lambda_i x_i \mid 0 \leq \lambda_i \leq 1, \sum_{i=1}^{r} \lambda_i = 1 \right\}$$

   (d.h. Q ist die konvexe Hülle von $x_1, \ldots, x_r$) und es existieren endlich viele Punkte $y_1, \ldots, y_s$, so dass

$$K = \left\{ \sum_{i=1}^{s} \mu_i y_i \mid \mu_i \geq 0 \right\}$$

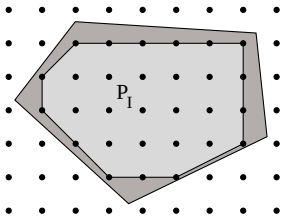


Q ist die konvexe Hülle der Eckpunkte von P. K entsteht aus den Strahlen, die zu unendlichen Kanten von P gehören. Man sagt, dass Q von $x_1, \ldots, x_r$ und K von $y_1, \ldots, y_s$ erzeugt werden.

3. Ist P ein Polyeder, so sei P_I die konvexe Hülle aller ganzzahligen Vektoren in P, d.h.

$$P_I = \left\{ \sum_{i=1}^{m} \lambda_i x_i \mid m \in \mathbb{N},\ \lambda_i \geq 0,\ \sum_{i=1}^{m} \lambda_i = 1,\ x_i \in P \cap \mathbb{Z}^n \right\}$$

Klar: P_I ist konvex.



4. Seien A, b rational. (Dann heißt $P = \{x \in \mathbb{Q}^n \mid Ax \leq b\}$ rationales Polyeder.)
Dann gilt: $\max\{c^T x \mid x \in P \cap \mathbb{Z}^n\} = \max\{c^T x \mid x \in P_I\}$

Satz (Meyer)
Ist P ein rationales Polyeder, so ist P_I ebenfalls ein (rationales) Polyeder.

Beweisidee (vgl. Abbildung 3.3):
$P = Q + K$ wobei Q von $x_1, \ldots, x_r$ und K von $y_1, \ldots, y_s$ erzeugt werden (vgl. 2.).

$$B = \left\{ \sum_{i=1}^{s} \mu_i y_i \mid 0 \leq \mu_i \leq 1 \right\}$$

$\widetilde{Q} = (Q + B)_I$ Polytop. Dann ist $P_I = \widetilde{Q} + K$.
(Genauer Beweis z.B. in [Sch99], Theorem 16.1)

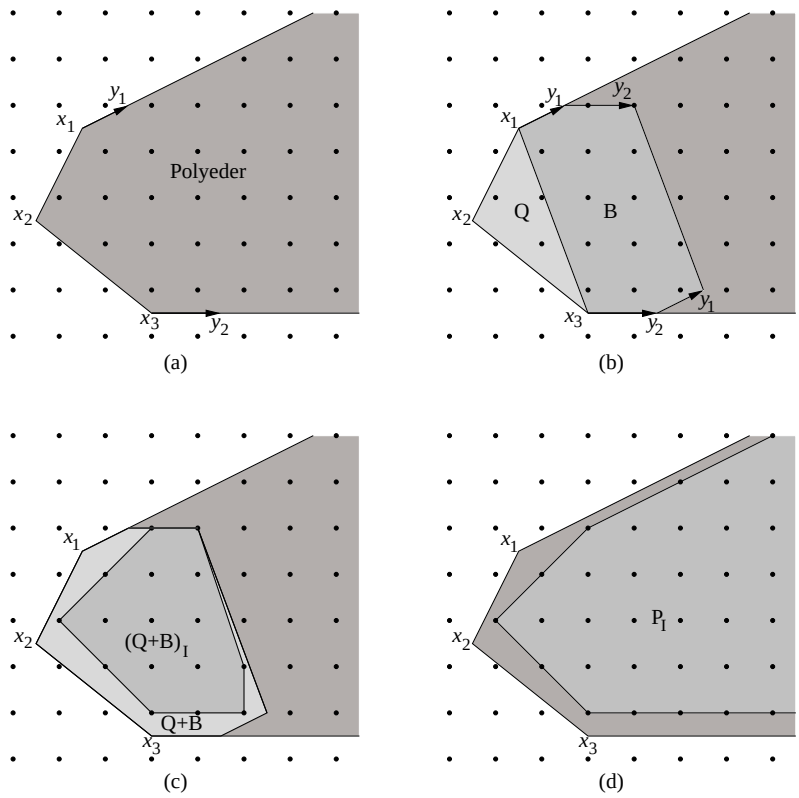


Abbildung 3.3: Graphische Darstellung zur Beweisidee des Satzes von Meyer

Korollar
Jedes ILP lässt sich als LP schreiben:

$$\max\{c^T x \mid Ax \leq b,\ x \in \mathbb{Z}^n\} = \max\{c^T x \mid \widetilde{A}x \leq \tilde{b},\ x \in \mathbb{Q}^n\}$$

Die Schwierigkeit ist dabei, die Ungleichungen $\widetilde{A}x \leq \tilde{b}$ zu finden!

5. **Satz (Karp, Papadimitriou)**
Ist $P = \{x \mid Ax \leq b\}$, A, b ganzzahlig, dann lassen sich $x_1, \ldots, x_t \in \mathbb{Z}^n$, $y_1, \ldots, y_u \in \mathbb{Z}^n$ bestimmen, so dass für die konvexe Hülle $\widetilde{Q}$ von $x_1, \ldots, x_t$ und den von $y_1, \ldots, y_u$ erzeugten Kegel K gilt:
$P_I = \widetilde{Q} + K$, wobei der Absolutbetrag der Komponenten dieser Vektoren beschränkt ist durch $(n+1)\Delta$ mit

$$\Delta = \max\{|D| \mid D \text{ ist Determinante einer Untermatrix von } (A|b)\}$$

Anmerkung: Die Ecken und Kanten des Polyeders erfüllen Gleichungen $A_t x = b_t$, wobei A_t, b_t gewisse Teilmatrizen/-vektoren von A bzw. b sind. Der Satz folgt dann letzten Endes aus der Cramer'schen Regel (genauer Beweis z.B. in [Sch99], Theorem 17.1).

Damit lässt sich dann Satz 3.9 beweisen.

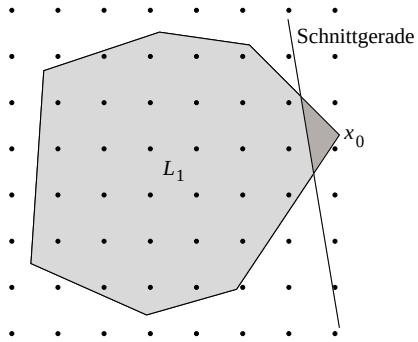
3.3 Schnittebenen-Verfahren (Cutting Planes Algorithm)

Allgemeine Idee
Sei das ILP gegeben durch $\max\{z(x) = c^T x \mid Ax \leq b,\ x \geq 0,\ x \in \mathbb{Z}^n\}$ (A, b rational/ganzzahlig) und die zugehörige LP-Relaxation durch $\max\{c^T x \mid Ax \leq b,\ x \geq 0,\ x \in \mathbb{Q}^n\}$.

$L = \{x \mid Ax \leq b,\ x \in \mathbb{Q}^n\}$ ist ein Polyeder.
Bestimme Ecke x_0 von L, die eine optimale Lösung der LP-Relaxation ist. Unterscheide nun folgende beiden Fälle:

1. $x_0 \in \mathbb{Z}^n$: fertig

2. $x_0 \notin \mathbb{Z}^n$: Bestimme eine weitere Ungleichung $a_{m+1}^T x \leq b_{m+1}$, die von x_0 nicht erfüllt wird, aber von jedem $x \in L \cap \mathbb{Z}^n$.
Betrachte nun $L_1 = \{x \mid Ax \leq b, a_{m+1}^T x \leq b_{m+1}, x \geq 0, x \in \mathbb{Q}^n\}$. Die optimale Lösung des ursprünglichen ILP liegt in L_1. Führe das Verfahren mit L_1 statt L durch und iteriere, bis (hoffentlich) eine ganzzahlige optimale Lösung erreicht wird.



Ein Verfahren zur Bestimmung solcher Schnitt- (Hyper-) Ebenen, bei dem eine ganzzahlige optimale Lösung nach endlich vielen Schritten erreicht wird stammt von Gomory (1958-1963).

BEISPIEL 3.3:
Dieses Beispiel dient der Beschreibung des Verfahrens. In Abbildung 3.4 sind die im Laufe der Berechnung erhaltenen Ergebnisse graphisch dargestellt.
Das ILP lautet: Maximiere $z(x_1, x_2) = x_1 + 25x_2$ bezüglich folgender Bedingungen:

$$\begin{aligned} -x_1 + 3x_2 &\leq 6 \\ 7x_1 + x_2 &\leq 35 \\ x_i &\geq 0,\ x_i \in \mathbb{Z},\ i = 1, 2 \end{aligned}$$

Löse zunächst die zugehörige LP-Relaxation mit Hilfe des Simplex-Algorithmus durch Einführung von Schlupfvariablen:

$$\begin{array}{rcrcrcrcl} -x_1 & + & 3x_2 & + & x_3 & & & = & 6 \\ 7x_1 & + & x_2 & & & + & x_4 & = & 35 \end{array} \qquad (*)$$

Bei $x^0 = (\frac{9}{2}, \frac{7}{2}, 0, 0)^T$ wird die Zielfunktion z maximal. (Es ist $z(x^0) = 92$.)

$$A = \begin{pmatrix} -1 & 3 & 1 & 0 \\ 7 & 1 & 0 & 1 \end{pmatrix}$$

a^1, a^2 bilden die Basis zu x^0. Wandle mit Gauss $Ax = b$ nun in $\widetilde{A}x = \tilde{b}$ um, wobei

$$\widetilde{A} = \begin{pmatrix} 1 & 0 & * & * \\ 0 & 1 & * & * \end{pmatrix}$$

Wir erhalten:

$$\begin{array}{rcrcrcl} x_1 & & - & \frac{1}{22}x_3 & + & \frac{3}{22}x_4 & = & \frac{9}{2} \\ & x_2 & + & \frac{7}{22}x_3 & + & \frac{1}{22}x_4 & = & \frac{7}{2} \end{array}$$

Wähle eine der Gleichungen, in denen der Wert der Basisvariablen der optimalen Lösung nicht ganzzahlig ist. Wir wählen z.B. die zweite Gleichung.

Schreibe in dieser Gleichung alle rationalen Zahlen in der Form $s = \lfloor s \rfloor + s'$ mit $0 \leq s' < 1$. Wir erhalten also:

$$x_2 + \frac{7}{22}x_3 + \frac{1}{22}x_4 = 3 + \frac{1}{2}$$

Bringe alle ganzzahligen Terme auf eine Seite:

$$x_2 - 3 = \frac{1}{2} - \frac{7}{22}x_3 - \frac{1}{22}x_4$$

Bei ganzzahligen Lösungen ist die linke Seite ganzzahlig. Daher muss in diesem Fall auch die rechte Seite ganzzahlig sein. Für nicht-negative x_3, x_4 bedeutet dies, dass die rechte Seite einen der Werte $0, -1, -2, \ldots$ annehmen muss. D.h. für ganzzahlige Lösungen gilt (Gomory-Schnitt):

$$\frac{1}{2} - \frac{7}{22}x_3 - \frac{1}{22}x_4 \leq 0 \qquad (**)$$

($(**)$ ist äquivalent zu $x_2 \leq 3$)
$(**)$ wird von allen $x \in L \cap \mathbb{Z}^n$ erfüllt, aber nicht von x^0.

Betrachte nun neues System (mit Schlupfvariablen):

$$\begin{array}{rcrcrcrcrcl} -x_1 & + & 3x_2 & + & x_3 & & & & & = & 6 \\ 7x_1 & + & x_2 & & & + & x_4 & & & = & 35 \\ & & & - & 7x_3 & - & x_4 & + & x_5 & = & -11 \end{array} \qquad (***)$$

Die neue hinzugekommene Zeile 3 ergibt sich aus $(**)$ durch Multiplikation mit 22.
Als optimale Lösung der LP-Relaxation dieses Systems liefert der Simplex-Algorithmus $x^1 = (\frac{32}{7}, 3, \frac{11}{7}, 0, 0)^T$. (Es ist $z(x^1) = 79\frac{4}{7}$.)
a^1, a^2, a^3 bilden die Basis zu x^1. Mit Hilfe der Gauss-Transformation erhalten wir

$$\begin{array}{rcrcrcrcl} x_1 & & & + & \frac{1}{7}x_4 & - & \frac{1}{154}x_5 & = & \frac{32}{7} \\ & x_2 & & & & + & \frac{1}{22}x_5 & = & 3 \\ & & x_3 & + & \frac{1}{7}x_4 & - & \frac{1}{7}x_5 & = & \frac{11}{7} \end{array}$$

Hier stehen nun die Gleichungen 1 und 3 zur Auswahl. Wir wählen die erste Gleichung und zerlegen die nicht-ganzzahlige Koeffizienten in einen ganzzahligen und einen positiven rationalen Teil:

$$x_1 + \frac{1}{7}x_4 - x_5 + \frac{153}{154}x_5 = 4 + \frac{4}{7}$$

Wir bringen nun die rationalen Anteile auf eine Seite und erhalten:

$$x_1 - x_5 - 4 = \frac{4}{7} - \frac{1}{7}x_4 - \frac{153}{154}x_5$$

Daraus ergibt sich dann der zweite Gomory-Schnitt:

$$\frac{4}{7} - \frac{1}{7}x_4 - \frac{153}{154}x_5 \leq 0$$

(Wegen $(*)$ und $(***)$ ist dies äquivalent zu $x_1 + 22x_2 \leq 70$.)
Man erhält nun das neue System

$$\begin{array}{rcrcrcrcrcrcr}
-x_1 & + & 3x_2 & + & x_3 & & & & & & & = & 6 \\
7x_1 & + & x_2 & & & + & x_4 & & & & & = & 35 \\
 & & & & -7x_3 & - & x_4 & + & x_5 & & & = & -11 \\
 & & & & & & -22x_4 & - & 153x_5 & + & x_6 & = & -88
\end{array}$$

Die LP-Relaxation liefert die optimale Lösung $x^2 = (4, 3, 1, 4, 0, 0)$. Diese ist ganzzahlig. Also ist die optimale Lösung des ursprünglichen ILP $x^* = (4, 3)$ mit $z(x^*) = 79$.

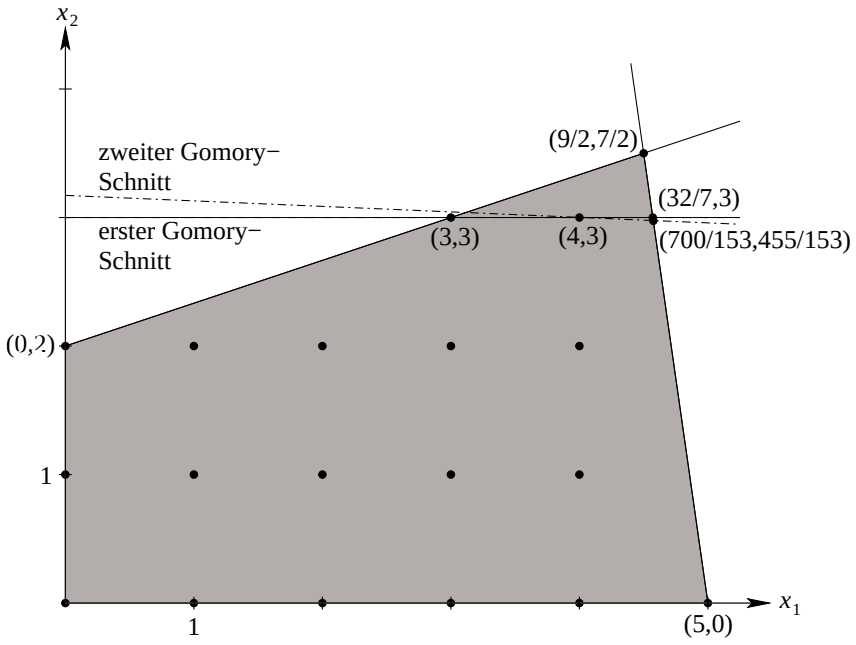


Abbildung 3.4: Graphische Veranschaulichung des Schnittebenen-Verfahrens

Man kann zeigen:
Bei geeigneter Wahl der Basen beim Simplex-Algorithmus und geeigneter Wahl der Gleichungen für die Gomory-Schnitte, findet das Verfahren nach endlich vielen Schritten eine optimale Lösung des ILP (für Details vgl. [Sch99], S.354-359).

3.4 Der Lenstra-Algorithmus

Im Jahre 1983 bewies H.W. Lenstra jr. den folgenden Satz, der für ziemliches Aufsehen sorgte:

3.13 Satz (H.W. Lenstra jr.)
Für festes $n \geq 1$ kann ILP

$$\max\{c^T x \mid Ax \leq b,\ x \in \mathbb{Z}^n\} \qquad (A, b \text{ rational})$$

in polynomialer Zeit gelöst werden.

Wesentlicher Bestandteil des zugrunde liegenden Algorithmus ist eine Reduktion des ILP auf ein Problem, „kurze“ Vektoren in ganzzahligen Gittern zu bestimmen. (Ein ganzzahliges Gitter im $\mathbb{R}^n$ ist von der Form $\{\sum_{i=1}^{n} a_i v_i \mid a_i \in \mathbb{Z}\}$, wobei $v_1, \ldots, v_n$ irgendeine Basis des $\mathbb{R}^n$ ist.) Hierfür wurde 1982 von A.K. Lenstra, H.W. Lenstra jr. und L. Lovasz (in Zusammenhang mit der Faktorisierung von Polynomen in $\mathbb{Q}[x]$) ein polynomialer Algorithmus angegeben.
Der Satz von Lenstra hat bisher keine praktische Bedeutung, ist aber von großem theoretischen Interesse, da mit seiner Hilfe für eine Reihe anderer Probleme gezeigt werden konnte, dass sie polynomial lösbar sind.

Kapitel 4

Optimierungsprobleme auf Graphen

4.1 Graphentheoretische Grundlagen

Ein *Graph* $G = (V, E)$ wird beschrieben durch zwei Mengen V und E, $V \cap E = \emptyset$ wobei V die *Eckenmenge* (engl. Vertices) und E die *Kantenmenge* (engl. Edges) darstellt, und einer Abbildung, die jeder Kante eine Menge $\{x, y\}$ (*ungerichteter* Graph) bzw. ein geordnetes Paar (x, y) (*gerichteter* Graph) von Ecken zuordnet.

Wir betrachten hier nur einfache (schlichte) Graphen, d.h. es existieren im Graph keine Mehrfachkanten zwischen zwei Ecken und keine Schleifen (d.h. Anfangs- und Endpunkt einer Kante sind identisch). Dann lassen sich Kanten durch die ihnen zugeordneten Ecken beschreiben:

$$\begin{array}{ll} \text{im ungerichteten Fall:} & E \subseteq \{\{x, y\} \mid x, y \in V,\ x \neq y\} \\ \text{im gerichteten Fall:} & E \subseteq \{(x, y) \mid x, y \in V,\ x \neq y\} \end{array}$$

Ein gerichteter oder ungerichteter Graph $G = (V, E)$ mit Gewichtsfunktion $w : E \to \mathbb{R}$ wird als *gewichteter Graph* oder als *Netzwerk* bezeichnet.

Jede Folge $F = (\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_n, v_{n+1}\})$ von Kanten eines ungerichteten Graphen $G = (V, E)$ heißt *Kantenzug* der Länge $l(F) = n$. Ist $v_1 = v_{n+1}$, so spricht man von einem geschlossenen Kantenzug, andernfalls von einem offenen. Als *einfachen Kantenzug* bezeichnet man einen Kantenzug, bei dem alle Kanten verschieden sind. Ein einfacher Kantenzug heißt *Weg*, falls $v_1, v_2, \ldots, v_n$ paarweise verschieden sind. Ein geschlossener Weg aus mindestens zwei Ecken wird als *Kreis* bezeichnet.
Kantenzüge, Wege und Kreise werden auf gerichteten Graphen analog definiert.

$a, b \in V$ heißen *verbindbar*, genau dann wenn mindestens ein Kantenzug von a nach b existiert. Der Abstand von a nach b ist definiert als

$$\text{dist}(a, b) = \begin{cases} \infty, \text{ falls } a \text{ und } b \text{ nicht verbindbar} \\ \min\{l(\text{Kantenzug von } a \text{ nach } b)\}, \text{ sonst} \end{cases}$$

Sei $G = (V, E)$ ein ungerichteter Graph.
G heißt *zusammenhängend*, genau dann wenn je zwei Ecken aus V verbindbar sind. Als Zusammenhangskomponente wird ein maximal zusammenhängender Subgraph G' von G bezeichnet.

Kreisfreie Graphen heißen *Wald*. Ein zusammenhängender Wald heißt (gerichteter oder ungerichteter) *Baum*.
Ein gerichteter Baum besitzt genau eine Ecke ohne eingehende Kante (*Wurzel*). Alle anderen Ecken haben genau eine eingehende Kante.

4.1 Satz

a) $G = (V, E)$ ungerichtet mit $|V| = n$. G ist ein Baum, genau dann wenn G zusammenhängend ist und $|E| = n - 1$.
(BEWEIS siehe [Jun02], Kapitel 1.2.8)

b) Ist G ein Wald mit k vielen Zusammenhangskomponenten, so hat G insgesamt $n - k$ viele Kanten.

Darstellungsmöglichkeiten von Graphen
Es gibt verschiedene „Datenstrukturen“ mit Hilfe derer sich Graphen darstellen lassen. Hierzu zählen

1. *Kantenlisten*

2. *Adjazenzlisten* (unsere Wahl der Darstellung)
Graph $G = (V, E)$ mit $|V| = n$.
$A_1, \ldots, A_n$, wobei A_i die Punkte j enthält mit $\{i, j\} \in E$ (bzw. $(i, j) \in E$).

3. *Adjazenzmatrizen*
Graph $G = (V, E)$ mit $|V| = n$.
$n \times n$-Matrix (a_{ij}), wobei

$$a_{ij} = \begin{cases} 1, & \text{falls } \{i, j\} \text{ (bzw. } (i, j)) \in E \\ 0, & \text{sonst} \end{cases}$$

4.2 Graph Scanning Algorithm (BFS)
BFS = Breadth First Search
Input: Gerichteter oder ungerichteter Graph $G = (V, E)$ und eine Ecke $s \in V$.
Output: Menge R der Ecken, die von s aus erreichbar sind und $T \subseteq E$, so dass (R, T) ein Baum bzw. ein gerichteter Baum mit Wurzel s ist (BFS-Baum).

Algorithmus:

(1) $R := \{s\}$, $Q := \{s\}$ (implementiert als FIFO-Schlange), $T := \emptyset$.

(2) Ist $Q = \emptyset$, fertig. Andernfalls wähle erstes Element $v \in Q$.

(3) Wähle $w \in V \setminus R$, wobei $\{v, w\}$ (bzw. (v, w)) $\in E$. Falls kein solches w exitiert, so setze $Q := Q \setminus \{v\}$ und gehe nach (2).

(4) Setze $R := R \cup \{w\}$, $Q := Q \cup \{w\}$, $T := T \cup \{e\}$, wobei $e = \{v, w\}$ bzw. $e = (v, w)$. Gehe anschließend nach (2).

4.3 Bemerkung

a) Der in 4.2 beschriebene Algorithmus liefert (bei mehrfacher Anwendung) die Ecken der Zusammenhangskomponenten eines ungerichteten Graphen.

b) Der in 4.2 beschriebene Algorithmus liefert für jedes $v \in V$, das von s aus erreichbar ist einen kürzesten Kantenzug zwischen s und v. Grund hierfür ist die durchgeführte Breitensuche (BEWEIS siehe [KV02], 2.18).

c) Sei der Graph G gegeben durch $G = (V, E)$ mit $|V| = n$ und $|E| = m$, dann besitzt der in 4.2 beschriebene Algorithmus eine Komplexität von $O(m + n)$.

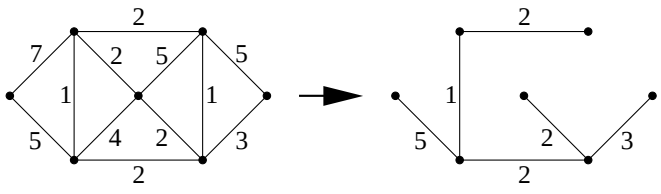
4.2 Minimale aufspannende Bäume (Minimum Spanning Trees)

4.4 Definition

a) Als *aufspannender Wald* eines ungerichteten Graphen $G = (V, E)$ wird ein Teilgraph $H = (V, T)$ bezeichnet, falls dieser ein Wald ist und die selben Zusammenhangskomponenten wie G besitzt.
Ist G zusammenhängend, dann ist H ein *aufspannender Baum*.
(Beachte, dass der in 4.2 beschriebene Algorithmus einen aufspannenden Wald liefert.)

b) Sei (G, w) ein ungerichtetes Netzwerk. Als *minimaler aufspannender Wald* wird ein aufspannender Wald (V, T) bezeichnet mit $w(T) := \sum_{e \in T} w(e)$ minimal.

BEISPIEL 4.1:
Folgende Abbildung zeigt ein Netzwerk, sowie einen zugehörigen aufspannenden Baum T mit $w(T) = 15$. Dabei handelt es sich allerdings nicht um einen minimalen aufspannenden Baum.



4.5 Algorithmus von Kruskal (1956)
Sei (G, w) ein ungerichtetes Netzwerk. Ziel ist die Bestimmung eines minimal aufspannenden Waldes.

(1) Sortiere Kanten in E in aufsteigender Reihenfolge nach ihrem Gewicht. Es gilt dann

$$E = \{e_1, e_2, \ldots, e_m\}, \text{ wobei } w(e_1) \leq w(e_2) \leq \ldots \leq w(e_m)$$

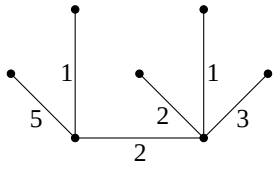
(2) $T := \emptyset$, $k := 1$

(3) Ist $(V, T \cup \{e_k\})$ ein Wald, so $T := T \cup \{e_k\}$.

(4) Ist $k = m$, fertig. Andernfalls $k := k + 1$ und gehe nach (3).

Bei dem hier beschriebenen Algorithmus handelt es sich um einen *gierigen Algorithmus* (greedy algorithm).

Angewandt auf das Netzwerk in Beispiel 4.1 liefert der Kruskal-Algorithmus z.B. folgenden minimal aufspannenden Baum T mit $w(T) = 14$.

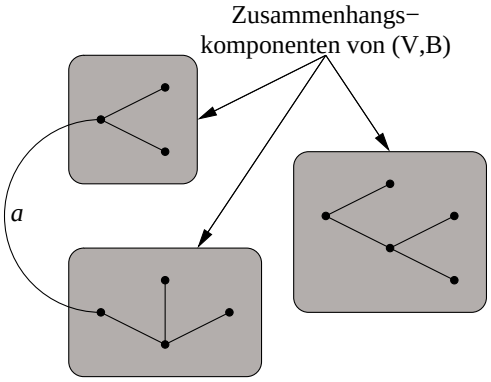


Die Gültigkeit des in 4.5 beschriebenen Algorithmus basiert auf folgendem Satz:

4.6 Satz
Sei $G = (V, E)$ ein endlicher Graph, $\mathcal{T} = \{A \subseteq E \mid (V, A) \text{ ist ein Wald}\}$. Dann gilt für $A, B \in \mathcal{T}$:

$$\text{Ist } |A| = |B| + 1 \text{ so existiert } a \in A \setminus B \text{ mit } B \cup \{a\} \in \mathcal{T}.$$

BEWEIS.
Nach Satz 4.1b) hat (V, B) genau eine Zusammenhangskomponente mehr als (V, A). (*Anmerkung:* Beachte, dass dabei lediglich die Anzahl betrachtet wird.) Daher gibt es mindestens eine Kante $a \in A \setminus B$, die zwei verschiedene Zusammenhangskomponenten von (V, B) verbindet (vgl. Abbildung). Dann ist $(V, B \cup \{a\})$ wieder ein Wald.



■

4.7 Gültigkeit des Kruskal-Algorithmus aus 4.5
Klar ist, dass (V, T) aus 4.5 ein aufspannender Wald ist. Nach Satz 4.1b) haben alle aufspannenden Wälder von $G = (V, E)$ gleich viele Kanten.

Sei $|T| = r$, $T = \{f_1, \ldots, f_r\}$ mit $w(f_1) \leq \ldots \leq w(f_r)$.
Sei (V, U) ein minimaler aufspannender Wald mit $U = \{g_1, \ldots, g_r\}$ mit $w(g_1) \leq \ldots \leq w(g_r)$. Angenommen es existiert ein Index i mit $w(g_i) < w(f_i)$. Sei i der kleinste solche Index.
Wende Satz 4.6 auf $A = \{g_1, \ldots, g_i\}$, $B = \{f_1, \ldots f_{i-1}\}$ an. Dann existiert $g_j \in A \setminus B$, so dass $(V, B \cup \{g_j\})$ ein Wald ist.
Dann ist $w(g_j) \leq w(g_i) < w(f_i)$. Dies ist ein Widerspruch zur Wahl von f_i im Algorithmus aus 4.5.
Somit gilt $w(g_k) \geq w(f_k)$ für alle k. Also $w(T) \leq w(U)$. Da (V, U) minimaler aufspannender Wald, muss Gleichheit gelten.

4.8 Bemerkung

a) Komplexität des in 4.5 beschriebenen Algorithmus
Betrachten die Schritte im Einzelnen:

(1) $O(m \log m)$ – sortieren der Kantenliste z.B. mittels Merge Sort

(2) $O(1)$

(3) $O(mn)$ – m-facher Test auf Kreis in Graph mit n Ecken (Dies kann z.B. durch Anwendung des in 4.2 beschriebenen Algorithmus auf $(V, T \cup \{e_k\})$ geprüft werden. Falls der BFS-Wald eine Kante von $T \cup \{e_k\}$ nicht enthält, so ist $T \cup \{e_k\}$ kein Wald und umgekehrt.)

(4) $O(1)$

Als Gesamtkomplexität ergibt sich also $O(mn)$. Bei geschickter Buchführung in (3) kann die Komplexität des Algorithmus auf $O(m \log n)$ gesenkt werden (vgl. [KV02], Theorem 6.4).

b) Mit Hilfe des Kruskal-Algorithmus lassen sich auch maximale aufspannende Wälder bestimmen. Dazu wendet man einfach den Algorithmus auf eine absteigend geordnete Kantenmenge an (gleichwertig wäre eine Ersetzung aller Kantengewichte $w(e)$ durch $-w(e)$).

c) Frage: Wann funktionieren solche gierigen Algorithmen? Welche Voraussetzungen müssen erfüllt sein?
Sei E endliche Menge, $\mathcal{T} \neq \emptyset$ Menge von Teilmengen von E. Es gelte:

(a) $A, B \subseteq E,\ A \in \mathcal{T},\ B \subseteq A \Rightarrow B \in \mathcal{T}$

(b) Austauscheigenschaft

$$A, B \in \mathcal{T},\ |A| = |B| + 1 \Rightarrow \exists a \in A \setminus B :\ B \cup \{a\} \in \mathcal{T}$$

Dann heißt $(E, \mathcal{T})$ *Matroid*.

Typisches Beispiel für Matroide:
E = Vektoren eines n-dimensionalen Vektorraums über $\mathbb{F}_p$
$\mathcal{T} = \{U \subseteq E \mid U \text{ linear unabhängig}\}$
$(E, \mathcal{T})$ ist dann ein Matroid. Bedingung (b) entspricht dem Steinitz'schen Austauschsatz. Die maximalen Elemente von $\mathcal{T}$ sind die Basen.

Man kann zeigen:

4.9 Satz
Genau dann liefert der Greedy-Algorithmus (analog 4.5) für jede Gewichtsfunktion $w : E \to \mathbb{R}$ eine in $\mathcal{T}$ maximale Menge $T \in \mathcal{T}$ (d.h. ist $U \subseteq E$, $T \subseteq U$, $U \in \mathcal{T} \Rightarrow T = U$) deren Gesamtgewicht $w(T) = \sum_{e \in T} w(e)$ minimal (bzw. maximal) ist, wenn $(E, \mathcal{T})$ ein Matroid ist.
(vgl. [Jun02], 4.6 und 4.7)

Anmerkung: Wenn wir im Kruskal-Algorithmus die Aussage „$(V, T \cup \{e_k\})$ ist ein Wald“ in Schritt (3) durch „$T \cup \{e_k\} \in \mathcal{T}$“ ersetzen, so zeigt das Argument von 4.7 (mit der entsprechenden Modifizierung), dass für Matroide der Greedy-Algorithmus funktioniert. Dies ist der Beweis für die eine Richtung im Satz 4.9.

Beispiel 4.2:
Anwendung von Bemerkung 4.8c): Problem der Ablaufplanung
$\mathcal{J}$ = Menge der Jobs, die auf einer Maschine ausgeführt werden sollen.
Jeder Job benötigt einen Tag. Außerdem gibt es zu jedem Job i einen Fertigstellungstermin $f(i)$, sowie eine Strafe $s(i)$, die bei Terminüberschreitung zu entrichten ist.
Gesucht ist nun eine Reihenfolge der Jobs, so dass die Summe der fälligen Strafen minimal wird. D.h. gesucht ist $T \subseteq \mathcal{J}$, wobei alle Jobs $i \in T$ rechtzeitig beendet werden können und $s = \sum_{i \in T} s(i)$ maximal.

Algorithmus:
Ordne die Jobs aufsteigend nach den zugehörigen Strafen. Gehe durch diese Liste, und weise einen Job i nur dann zurück, wenn er zusammen mit den zuvor akzeptierten Jobs nicht pünktlich ausgeführt werden kann.
Dies liefert uns:

$$\mathcal{T} = \{T \mid T \subseteq \mathcal{J},\ \forall n \in \mathbb{N} :\ |\{j \in T \mid f(j) \leq n\}| \leq n\}$$

Zahlenbeispiel:

Job i	$f(i)$	$s(i)$	Ausführungsnummer
1	1	10	1
2	1	9	zurückgewiesen
3	3	7	3
4	2	6	2
5	3	4	zurückgewiesen
6	6	2	4

4.3 Kürzeste Wege

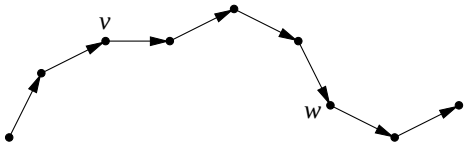
Problem:
Gegeben ein gerichtetes oder ungerichtetes Netzwerk G mit Gewichtsfunktion $w :\ E(G) \to \mathbb{R}$, $s, t \in V(G)$.

Gesucht ist ein Weg (bzw. einfacher Kantenzug) von minimalem Gewicht von s nach t. Ist W ein solcher Weg mit $w(W) = \sum_{e \in E, e \in W} w(e)$, so sprechen wir von einem *kürzesten Weg*.

4.10 Bemerkung
Das Finden kürzester Wege in einem Netzwerk ist im allgemeinen ein schwieriges Problem.
Ein Algorithmus zur Bestimmung eines Weges mit minimalem Gewicht sei gegeben. Sei $w(e) = -1$ für alle $e \in E(G)$. Falls ein (gerichteter) Weg über alle Ecken von G von s nach t existiert ($s \neq t$ Ecken) – ein sog. Hamilton'scher s-t-Weg –, so ist dies ein Weg von Gewicht $1 - |V(G)|$. Wege von s nach t von kleinerem Gewicht gibt es nicht.
Der Algorithmus liefert also eine Möglichkeit zu überprüfen, ob ein Graph einen Hamilton'schen Weg enthält. Dieses Problem ist aber NP-vollständig.

Das Problem vereinfacht sich, wenn man nur nicht-negative Gewichte zulässt (allgemeiner: keine Kreise mit negativem Gesamtgewicht).
Bei nicht-negativen Gewichtsfunktionen lässt sich das ungerichtete „kürzeste-Weg-Problem" auf das für gerichtete Graphen übertragen, indem man jede Kante durch zwei entgegengesetzte gerichtete Kanten von gleichem Gewicht wie die ursprüngliche Kante ersetzt. Wir werden uns im Folgenden nur mit gerichteten Netzwerken mit nicht-negativer Gewichtsfunktion befassen.

Bezeichnungen
Sei W ein gerichteter Weg, $v, w \in V(W)$, wobei v auf dem Weg „vor" w liegt. Dann bezeichnen wir mit $W_{[v,w]}$ den Teilweg von W von v nach w.



4.11 Algorithmus von Dijkstra (1959)
Input: Ein gerichtetes Netzwerk G mit Gewichtsfunktion $w : E(G) \to \mathbb{R}_{\geq 0}$, $s \in V(G)$.
Output: Kürzeste Wege von s zu allen $v \in V(G)$ und ihre Längen. Genauer gesagt: Zu jedem $v \in V(G)$ wird $l(v)$ und $p(v)$ ausgegeben, wobei $l(v)$ die Länge eines kürzesten s-v-Weges W ist. Dabei ist $W = W' \cup (p(v), v)$, wobei W' der kürzeste s-$p(v)$-Weg und $(p(v), v)$ die letzte Kante von W ist ($p(v)$ ist also die Vorgängerecke von v).

Algorithmus:

(1) $l(s) := 0$
$l(t) := \infty$ für alle $t \in V(G) \setminus \{s\}$
$R := \emptyset$

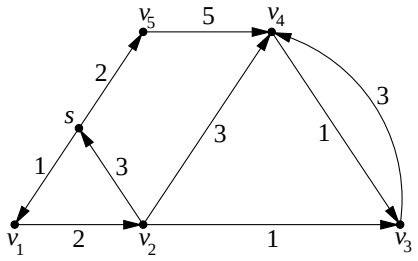
(2) Bestimme Ecke $t \in V(G) \setminus R$ mit $l(t) = \min_{v \in V(G) \setminus R} l(v)$

(3) $R := R \cup \{t\}$

(4) Für alle $v \in V(G) \setminus R$ mit $(t, v) \in E(G)$: Falls $l(v) > l(t) + w((t, v))$, so $l(v) := l(t) + w((t, v))$, $p(v) := t$.

(5) Ist $R \neq V(G)$, so gehe nach (2).

BEISPIEL 4.3:
Gegeben ist folgendes Netzwerk:



Gesucht ist der kürzeste Weg zwischen den Knoten s und v_4.

Die folgende Berechnungstabelle zeigt schrittweise das Vorgehen des Dijkstra-Algorithmus:

Schritt	Knoten v mit $(l(v), p(v))$						Menge R
	s	v_1	v_2	v_3	v_4	v_5	
0	$(0, -)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$\{\}$
1	$(0, -)$	$(1, s)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$(2, s)$	$\{s\}$
2	$(0, -)$	$(1, s)$	$(3, v_1)$	$(\infty, -)$	$(\infty, -)$	$(2, s)$	$\{s, v_1\}$
3	$(0, -)$	$(1, s)$	$(3, v_1)$	$(\infty, -)$	$(7, v_5)$	$(2, s)$	$\{s, v_1, v_5\}$
4	$(0, -)$	$(1, s)$	$(3, v_1)$	$(4, v_2)$	$(6, v_2)$	$(2, s)$	$\{s, v_1, v_5, v_2\}$
5	$(0, -)$	$(1, s)$	$(3, v_1)$	$(4, v_2)$	$(6, v_2)$	$(2, s)$	$\{s, v_1, v_5, v_2, v_3\}$
6	$(0, -)$	$(1, s)$	$(3, v_1)$	$(4, v_2)$	$(6, v_2)$	$(2, s)$	$\{s, v_1, v_5, v_2, v_3, v_4\}$

Der kürzeste s-v_4-Weg kann durch „Rückverfolgung“ der $p(v)$-Knoten ausgehend von v_4 gefunden werden. Dies ist also s-v_1-v_2-v_4 und besitzt die Länge 6.

4.12 Gültigkeit des Dijkstra-Algorithmus aus 4.11
Zeige, dass bei jedem Durchlauf von (2) folgendes gilt:

(a) Für alle $t \in R$ und alle $v \in V(G) \setminus R$: $l(t) \leq l(v)$

(b) Für alle $t \in R$: $l(t)$ ist die Länge des kürzesten s-t-Weges in G. Ist $l(t) < \infty$, so existiert ein kürzester s-t-Weg, dessen letzte Kante $(p(t), t)$ ist (außer wenn $t = s$) und dessen sämtliche Ecken zu R gehören.

(c) Für alle $v \in V(G) \setminus R$: $l(v)$ ist die Länge des kürzesten Weges von s nach v in $G[R \cup \{v\}]$ (dieser Graph besteht aus den Ecken in $R \cup \{v\}$ sowie allen Kanten in $E(G)$, die zwischen Ecken in $R \cup \{v\}$ existieren). Ist $l(v) < \infty$, so ist $p(v) \in R$ und $l(v) = l(p(v)) + w((p(v), v))$.

Ist $R = V(G)$, so folgt aus (b) die Gültigkeit des Dijkstra-Algorithmus.

Es ist trivial zu zeigen, dass (a), (b) und (c) nach Schritt (1) des Algorithmus gelten.

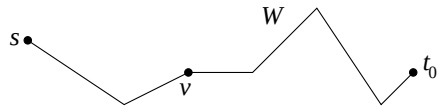
Zeige nun, dass wenn (a), (b) und (c) vor der Ausführung von (3) und (4) galten, dass diese auch danach noch gelten.
Sei R_a die „alte“ R-Menge und t_0 die Ecke, die in (2) gewählt wird. In (3) wird dann die „neue“ R-Menge durch Hinzufügen von t_0 zu R_a erzeugt: $R_n = R_a \cup \{t_0\}$

1. Zeige, dass (3) und (4) die Eigenschaft (a) erhalten:
   Für alle $x \in R_n$ und alle $y \in V(G) \setminus R_n$ gilt:

   $$\begin{array}{rcll} l(x) & \leq & l(t_0) & \text{nach (a) für } x \neq t_0 \\ & \leq & l(y) & \text{nach Wahl von } t_0 \text{ in (2)} \end{array}$$

   Also gilt (a) auch nach Durchführung von Schritt (3) und (4).

2. Zeige, dass (3) und (4) die Eigenschaft (b) erhalten:
   Da (b) vor der Ausführung von (3) und (4) gilt, bleibt lediglich zu überprüfen, ob das zu R hinzugekommene t_0 ebenfalls (b) erfüllt.
   Da (c) vor Schritt (3) galt, ist $l(t_0)$ die Länge des kürzesten Weges von s nach t_0 in $G[R_n]$. Wir müssen nun zeigen, dass es keinen kürzeren Weg von s nach t_0 gibt, der eine Ecke in $V(G) \setminus R_n$ enthält.
   Angenommen es gibt doch einen solchen Weg W mit $w(W) < l(t_0)$. Sei $v \notin R_n$ die erste Ecke auf W nach s, die in $V(G) \setminus R_n$ liegt.

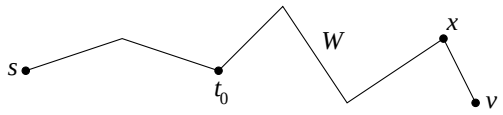
   

   Da (c) vor (3) galt, gilt $l(v) \leq w(W_{[s,v]})$. Da alle Gewichte nicht-negativ sind, folgt:

   $$w(W_{[s,v]}) \leq w(W) < l(t_0)$$

   Daraus folgt, dass $l(v) < l(t_0)$. Dies ist ein Widerspruch zur Wahl von t_0 in (2). Schritt (3) erhält also die Eigenschaft (b). Ebenso erhält Schritt (4) diese Eigenschaft, da dabei lediglich solche Ecken beeinflusst werden, die nicht in R_n liegen.

3. Zeige, dass (3) und (4) die Eigenschaft (c) erhalten:
   Zunächst: Für alle $v \in V(G) \setminus R$ existiert ein Weg von s nach v der Länge $l(v)$ in $G[R \cup \{v\}]$. (Dies gilt, weil (c) vor (3) und (4) galt und nach (3) und (4) $l(v)$ höchstens so verändert wird, dass diese Aussage auch gilt.)
   Angenommen (c) gilt nicht mehr nach der Ausführung von Schritt (3) und (4). Dann existiert ein $v \in V(G) \setminus R_n$ und ein Weg W von s nach v in $G[R_n \cup \{v\}]$, der kürzer als $l(v)$ ist. Da vor (3) und (4) die Eigenschaft (c) gültig war, muss W durch t_0 gehen.



(x, v) sei die letzte Kante von W (evtl. mit $x = t_0$). Dann ist $x \in R_n$. Nach (a) ist $l(x) \leq l(t_0)$ und nach (4) ist $l(v) \leq l(x) + w((x, v))$.
Also ist

$$\begin{aligned} l(v) &\leq l(x) + w((x,v)) \\ &\leq l(t_0) + w((x,v)) \\ &\leq w(W_{[s,t_0]}) + w((x,v)) \\ &\leq w(W) \\ &< l(v), \end{aligned}$$

wobei die Ungleichung in der dritten Zeile gilt, da nach (b), das nach (3) und (4) gilt, $l(t_0)$ die Länge des kürzesten s-t_0-Wegs in G ist. Der Widerspruch $l(v) < l(v)$ zeigt, dass auch (c) nach der Ausführung von (3) und (4) erhalten bleibt.

4.13 Bemerkung

a) Der Dijkstra-Algorithmus hat eine Komplexität von $O(n^2)$.

b) Der Dijkstra-Algorithmus beruht auf der einfach zu beweisenden Tatsache, dass es in einem gerichteten Netzwerk G mit nicht-negativer Gewichtsfunktion (allgemeiner: ohne Kreise von negativem Gesamtgewicht) gilt: Sind $s, v \in V(G)$, (t, v) letzte Kante eines kürzesten Weges W von s nach v, so ist $W_{[s,t]}$ ein kürzester Weg von s nach t (Bellman-Prinzip).

c) In ungerichteten Graphen mit *ganzzahligen* nicht-negativen Gewichten gibt es einen $O(n)$-Algorithmus für das „Kürzester-Weg-Problem“ (vgl. [Tho99]).

d) Der in 4.11 beschriebene Algorithmus lässt sich verallgemeinern, falls die Gewichtsfunktion so beschaffen ist, dass es keine Kreise von negativem Gesamtgewicht gibt.
Ein solcher Algorithmus wurde von Moore, Bellman und Ford entwickelt und besitzt eine Komplexität von $O(mn)$ (vgl. [KV02], Abschnitt 7.1).

4.4 Das Traveling-Salesman-Problem (TSP)

Sei K_n der vollständige Graph auf n Ecken.
Problem (TSP)
Input: Gewichtsfunktion $w : E(K_n) \to \mathbb{Q}_{\geq 0}$ auf Kanten des vollständigen Graphen K_n.
Output: Kürzester geschlossener Weg, der alle Ecken von K_n enthält (d.h. Hamilton'scher Kreis mit minimalem Gewicht).
Das TSP ist NP-hart!

4.14 ILP-Formulierung des TSP

Das TSP lässt sich als ganzzahliges lineares Optimierungsproblem beschreiben:

$$V(K_n) = \{1, \ldots, n\},\ w_{ij} = w(\{i,j\})$$

$$x_{\{i,j\}} = \begin{cases} 1, & \text{falls Weg durch die Kante } \{i,j\} \text{ geht} \\ 0, & \text{sonst} \end{cases}$$

(*Anmerkung:* Wir schreiben im Folgenden statt $x_{\{i,j\}}$ meist x_{ij}, d.h., dass insbesondere $x_{ij} = x_{ji}$ gilt.)

Zielfunktion:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_{ij} \text{ minimal}$$

Bedingungen:

1. $x_{ij} \in \{0,1\}$
2. $\sum_{j \neq i} x_{ij} = 2$ für alle $i = 1, \ldots, n$. Dadurch erreicht man, dass bei der Lösung jede Ecke i mit genau zwei weiteren Ecken über Wegkanten verbunden ist.

Jetzt ist es immer noch möglich, dass die Lösung aus mehreren geschlossenen Kreisen (sog. Teiltouren) besteht. Daher ist eine weitere Bedingung notwendig:

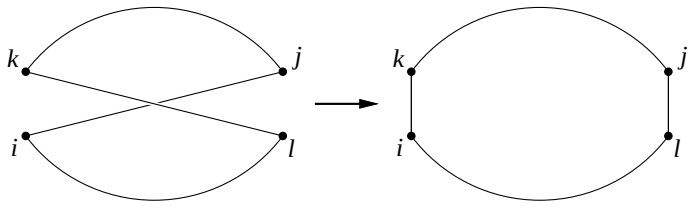
3. Für alle $S \subseteq \{1, \ldots, n\}$ mit $3 \leq |S| \leq \lfloor \frac{n}{2} \rfloor$:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 2$$

Problem:
Exponentielles Anwachsen der Bedingungsanzahl. Z.B. ergeben sich bereits für $n = 20$ ca. $\frac{2^{20}}{2} = 2^{19} = 524288$ Bedingungen.

4.15 Lokale Optimierungsverfahren

Gegeben ist ein geschlossener Weg durch alle Knoten.
Die einfachste *lokale Optimierungsmethode* besteht darin, dass zwei Kanten so durch zwei andere ersetzt werden, dass weiterhin ein geschlossener Weg existiert, dessen Gesamtgewicht nun aber kleiner ist als das seines „Vorgängers“.



Sind die Kanten verbunden wie in obiger Abbildung mit $w_{ij} + w_{kl} > w_{ik} + w_{jl}$, so ersetze im ursprünglichen Weg die Kanten $\{i, j\}$ und $\{k, l\}$ durch $\{i, k\}$ und $\{j, l\}$. Dieses Verfahren lässt sich auf mehrere Kanten verallgemeinern.

4.16 Schranken

a) Entferne aus einem Hamilton'schen geschlossenen Weg H eine Kante. Dies liefert einen aufspannenden Baum. Der Kruskal-Algorithmus berechnet das Gewicht eines minimalen aufspannenden Baumes T. Dann gilt $w(T) \leq w(H)$.
Ferner gilt: Wenn w die Dreiecksungleichung erfüllt (d.h. $w(a, b) \leq w(a, c) + w(c, b)$ für alle $a, b, c \in K_n$), dann ist $w(H) \leq 2w(T)$, falls H ein Hamilton'scher Kreis von minimalem Gewicht ist.

b) Wird das 2-Kanten-Optimierungsverfahren aus 4.15 iterativ angewandt, bis keine Verbesserung mehr auftritt, so führt dies zu geschlossenem Weg, dessen Länge nicht größer ist als das $4\sqrt{n}$-fache des Optimums (es lassen sich Ausgangssituationen konstruieren, in denen obiges Verfahren einen geschlossenen Weg liefert, dessen Gewicht um einen Faktor $\frac{1}{4}n^{1/4}$ schlechter ist, als das Gewicht der optimalen Lösung).

Kapitel 5

Das Knapsack-Problem

5.1 0-1-Knapsack-Probleme
Gegeben: $n, g_1, \ldots, g_n, c_1, \ldots, c_n, b \in \mathbb{N}_0$

Gesucht: $S \subseteq \{1, \ldots, n\}$, so dass

$$\sum_{j \in S} c_j \leq b \text{ und } \sum_{j \in S} g_j \text{ maximal}$$

Eine äquivalente Formulierung lautet:

$$\text{Maximiere } \sum_{j=1}^{n} g_j x_j \text{ bezüglich } \sum_{j=1}^{n} c_j x_j \leq b,\ x_j \in \{0,1\} \qquad (*)$$

Damit hat man eine ILP-Formulierung des 0-1-Knapsack-Problems.
Interpretation: g_j Gewinn für Ware j, Ware j benötigt c_j Einheiten einer gewissen Resource. (Frachtladungen, Investment-Planungen, etc.)
Ist in $(*)$ $x_j \in \mathbb{N}_0$ statt $x_j \in \{0,1\}$ „erlaubt“, so bezeichnet man dies als *ganzzahliges Knapsack-Problem.*

Die LP-Relaxation von $(*)$ (d.h. $x_j \in \mathbb{Q}$, $0 \leq x_j \leq 1$) hat eine einfach zu bestimmende Lösung:

5.2 Satz (Dantzig, 1957)
Seien $g_1, \ldots, g_n$ und $c_1, \ldots, c_n$ so nummeriert, dass $\frac{g_1}{c_1} \geq \frac{g_2}{c_2} \geq \ldots \geq \frac{g_n}{c_n}$. Sei $k = \min\left\{ j \mid \sum_{i=1}^{j} c_i > b \right\}$. Dann hat die LP-Relaxation von $(*)$ die optimale Lösung x^* mit

$$\begin{aligned} x_j^* &= 1, \text{ für } j = 1, \ldots, k-1 \\ x_k^* &= \frac{b - \sum_{i=1}^{k-1} c_i}{c_k} \\ x_j^* &= 0, \text{ für } j = k+1, \ldots, n \end{aligned}$$

BEWEIS.
Angenommen x^* ist nicht optimal. Sei stattdessen $(y_1, \ldots, y_n)$ eine optimale Lösung. Außerdem gälte o.B.d.A. $\sum_{i=1}^{n} c_i \geq b$. Dann ist klar, dass

$$\sum_{j=1}^{n} c_j y_j = b \text{ und}$$

$$\sum_{j=1}^{n} g_j y_j > \sum_{j=1}^{n} g_j x_j^*$$

Sei l der kleinste Index für den gilt $y_l \neq x_l^*$. Wähle nun unter allen optimalen Lösungen $(y_1, \ldots, y_n)$ diejenige aus, bei der dieses l maximal ist.
Dann ist $y_l < x_l^*$:
Klar, falls $l < k$. Ist $l = k$, so ist $y_j = 1$ für $j = 1, \ldots, k-1$, womit dann nach Konstruktion von x_k^* auch $y_k < x_k^*$ gilt. $l > k$ ist nicht möglich.

Ersetze nun y_l durch x_l^* und verkleinere die übrigen $y_{l+1}, \ldots, y_n$ so, dass die Schranke b erreicht wird. Dies liefert eine neue zulässige Lösung $(z_1, \ldots, z_n)$ mit $z_i = x_i^*$ für $i = 1, \ldots, l$ und $\sum_{j=l+1}^{n} c_j(y_j - z_j) = c_l(z_l - y_l)$. Nun:

$$\begin{aligned}
\sum_{j=1}^{n} g_j z_j &= \sum_{j=1}^{n} g_j y_j + \underbrace{(z_l - y_l)}_{>0} \frac{c_l g_l}{c_l} - \sum_{j=l+1}^{n} \underbrace{(y_j - z_j)}_{\geq 0} \frac{c_j g_j}{c_j} \\
&\geq \sum_{j=1}^{n} g_j y_j + \underbrace{\left[(z_l - y_l)c_l - \sum_{j=l+1}^{n} (y_j - z_j)c_j \right]}_{=0} \frac{g_l}{c_l} \\
&= \sum_{j=1}^{n} g_j y_j
\end{aligned}$$

Da $(y_1, \ldots, y_n)$ eine optimale Lösung, muss gelten:

$$\sum_{j=1}^{n} g_j z_j = \sum_{j=1}^{n} g_j y_j$$

Also ist auch $(z_1, \ldots, z_n)$ eine optimale Lösung. Dabei ist $z_i = x_i^*$ für $i = 1, \ldots, l$. Dies ist ein Widerspruch zur Wahl von l ∎

5.3 Bemerkung
Komplexität der Durchführung von 5.2:
$O(n \log n)$ für das Sortieren und $O(n)$ für die Bestimmung von k und das Setzen der x_j.
Insgesamt besitzt die LP-Relaxation des Knapsack-Problems also Komplexität $O(n \log n)$ (*Anmerkung:* Es existiert sogar ein $O(n)$-Algorithmus; vgl. [KV02], Corollary 17.5).

5.4 Heuristik für das 0-1-Knapsack-Problem
Gegeben: $(*)$, $\frac{g_1}{c_1} \geq \ldots \geq \frac{g_n}{c_n}$ (o.B.d.A. $c_i \leq b$ für $i = 1, \ldots, n$).
Bestimme k wie in 5.2. Unterscheide zwei Fälle:

1. $g_1 + \ldots + g_{k-1} > g_k$
   Setze $x_1 = \ldots = x_{k-1} = 1$, $x_k = 0$, $\tilde{b} = b - c_1 - \ldots - c_{k-1}$.

2. $g_1 + \ldots + g_{k-1} \leq g_k$
   Setze $x_1 = \ldots = x_{k-1} = 0$, $x_k = 1$, $\tilde{b} = b - c_k$.

Wende anschließend Rekursion auf

$$c_{k+1}x_{k+1} + \ldots + c_n x_n \leq \tilde{b},\ x_i \in \{0,1\}$$

$$g_{k+1}x_{k+1} + \ldots + g_n x_n \text{ maximal}$$

an.

5.5 Bemerkung
Die Heuristik aus 5.4 liefert eine zulässige Lösung für $(*)$. Sei z_H der Wert der Zielfunktion dieser Lösung und z^* der optimale Wert des Knapsack-Problems. Dann gilt $z^* \geq z_H \geq \frac{1}{2}z^*$.
Denn: Ist z_{LP} der optimale Wert der LP-Relaxation, so gilt

$$z^* \leq z_{LP} \leq \underbrace{g_1 + \ldots + g_{k-1}}_{\leq z_H} + \underbrace{g_k}_{\leq z_H} \leq 2z_H$$

5.6 Satz (Bellman, 1957, Dantzig, 1957)
Gegeben: $(*)$
Für $i = 1, \ldots, n$ und $d = 0, \ldots, b$ sei

$$z_i(d) = \max\left\{ \sum_{j=1}^{i} g_j x_j \mid \sum_{j=1}^{i} c_j x_j \leq d \right\},\ x_j \in \{0,1\}$$

Dann ist

$$z_1(d) = \begin{cases} 0, \text{ falls } c_1 > d \\ g_1, \text{ falls } c_1 \leq d \end{cases} \quad \text{für alle } d = 0, \ldots, b$$

und

$$z_i(d) = \begin{cases} z_{i-1}(d), \text{ falls } c_i > d \\ \max\{z_{i-1}(d),\ g_i + z_{i-1}(d - c_i)\}, \text{ falls } c_i \leq d \end{cases}$$

Beweis.
Trivial für $i = 1$.
Sei $i > 1$ und $\sum_{j=1}^{i} g_j x_j^* = z_i(d)$.
Ist $x_i^* = 0$, so ist $z_i(d) = z_{i-1}(d)$.
Ist $x_i^* = 1$, so ist $c_i \leq d$. Dann ist klar, dass $z_i(d) = g_i + z_{i-1}(d - c_i)$.
Damit folgt dann die Behauptung. ∎

5.7 Dynamischer Programmierungsalgorithmus für Knapsack
(Bellman, Dantzig)
Bestimme $z_n(b)$ rekursiv nach Satz 5.6. Dies ist dann der optimale Wert des 0-1-Knapsack-Problems. Zugehörige optimale Lösung $(x_1^*, \ldots, x_n^*)$ erhält man durch „Rückwärtsrekursion“ wie folgt:

$$x_n^* = \begin{cases} 0, \text{ falls } z_n(b) = z_{n-1}(b) \\ 1, \text{ sonst} \end{cases}$$

Für $i \leq n-1$ sei $b_i = b - \sum_{j=i+1}^{n} c_j x_j^*$.

$$x_i^* = \begin{cases} 0, \text{ falls } z_i(b_i) = z_{i-1}(b_i) \\ 1, \text{ sonst} \end{cases} \text{, wobei } z_0(b_1) = 0$$

Der Algorithmus besitzt Komplexität $O(nb)$ (Er ist also bei festem b polynomial: pseudo-polynomialer Algorithmus). Ein polynomialer Algorithmus für das 0-1-Knapsack-Problem ist nicht zu erwarten, da es NP-hart ist.

Beispiel 5.1:
Maximiere $16x_1 + 19x_2 + 23x_3 + 28x_4$ bezüglich $2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 7$, $x_i \in \{0, 1\}$.
Wir wenden Satz 5.6 sowie den dynamischen Programmieralgorithmus aus 5.7 an.

$$z_1(d) = \begin{cases} 0 & 0 \leq d \leq 1 \\ 16 & 2 \leq d \leq 7 \end{cases}$$

$$z_2(d) = \begin{cases} 0 & 0 \leq d \leq 1 \\ 16 & d = 2 \\ 19 & 3 \leq d \leq 4 \quad (\max(16, 19+0)) \\ 35 & 5 \leq d \leq 7 \quad (\max(16, 19+16)) \end{cases}$$

$$z_3(d) = \begin{cases} z_2(d) & 0 \leq d \leq 3 \\ 23 & d = 4 \quad (\max(19, 23+0)) \\ 35 & d = 5 \quad (\max(35, 23+0)) \\ 39 & d = 6 \quad (\max(35, 23+16)) \\ 42 & d = 7 \quad (\max(35, 23+19)) \end{cases}$$

$z_4(7) = \max\{z_3(7), 28 + z_3(2)\} = 44$. $x_4^* = 1$, $x_3^* = x_2^* = 0$, da $z_3(2) = z_2(2) = z_1(2)$, $x_1^* = 1$, da $z_1(2) > 0$.

Kapitel 6

Weitere Themen

Es gibt viele weitere Probleme im Bereich der diskreten Optimierung, die hier nicht explizit besprochen wurden. Einige davon wollen wir in diesem Kapitel allerdings noch kurz erwähnen und einen Verweis auf entsprechende Literatur geben.

1. *Matchings (Korrespondenzen)*
   Ein „Matching“ eines Graphen ist eine Menge von Kanten, von denen keine zwei einen Endpunkt gemeinsam haben.

   Hauptprobleme:

   - Bestimmung von Matchings maximaler Größe
   - Bestimmung von Matchings maximalen/minimalen Gewichts in gewichteten Graphen

   Anwendungen z.B. bei Zuordnungsproblemen.

   (vgl. [KV02], Kapitel 10 und 11)

2. *Matroide*
   Der in Bemerkung 4.8 angesprochene Begriff des „Matroids“ spielt eine wichtige Rolle in der kombinatorischen Optimierung. Die wichtigsten über den Satz 4.9 hinausgehenden Resultate findet man z.B. in [KV02], Kapitel 13 und 14.

3. *Flüsse in Netzwerken*
   Wege maximalen Gewichts (maximaler Kapazität) zwischen zwei ausgezeichneten Punkten in gerichteten Netzwerken.

   Grundlegender Algorithmus:
   Max-Flow-Min-Cut-Algorithmus von Ford und Fulkerson

   (vgl. [KV02], Kapitel 8)

4. *Netzwerk-Design, Steiner-Bäume*
   Ein typisches Problem in diesem Bereich ist die Bestimmung eines Netzwerks aus horizontalen und vertikalen Strecken von minimaler Länge, das eine gegebene Menge von Punkten in der Ebene verbindet (vgl. [KV02], Kapitel 20).

5. *Ablaufplanung*
   Hier geht es um Methoden, wie die Reihenfolge der Durchführung gewisser Jobs (Aufgaben) optimal zu planen ist. Eine wichtige Rolle spielen dabei die Anzahl der zur Verfügung stehenden Maschinen (Prozessoren, etc.) und Vorgaben von der Art, dass gewisse Jobs beendet sein müssen bevor andere begonnen werden können. Eine erste Einführung in Ein- und Mehrmaschinenprobleme findet man in [Ihr02], Kapitel VI.

Literaturverzeichnis

[Aig99] Martin Aigner, *Diskrete Mathematik*, Vieweg, 1999.

[CCPS97] Wiliam J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver, *Combinatorial Optimization*, Wiley, 1997.

[GLS93] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer, 1993.

[Ihr02] Thomas Ihringer, *Diskrete Mathematik*, Heldermann, 2002.

[Jun02] Dieter Jungnickel, *Graphs, Networks and Algorithms*, Springer, 2002.

[KV02] Bernhard Korte and Jens Vygen, *Combinatorial Optimizations*, Springer, 2002.

[NW99] George L. Nemhauser and Laurence A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, 1999.

[PS98] Christos H. Papadimitriou and Kenneth Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, 1998.

[Sch99] Alexander Schrijver, *Theory of Linear and Integer Programming*, Wiley, 1999.

[Tho99] Mikkel Thorup, *Undirected single-source shortest paths with positive integer weights in linear time*, Journal of the ACM **46** (1999), no. 3, 362–394.

[Wol98] Laurence A. Wolsey, *Integer Programming*, Wilsey, 1998.