Diskrete Optimierung

(Algorithmische Diskrete Mathematik II, kurz ADM II)

Skriptum zur Vorlesung im SS 2015

Prof. Dr. Martin Grötschel
Institut für Mathematik
Technische Universität Berlin

finale Version vom 20. Juli 2015

Vorwort

Bei dem vorliegenden Skript handelt es sich um die Ausarbeitung der vierstündigen Vorlesung „Algorithmische Diskrete Mathematik II (Diskrete Optimierung)“ (mit zugehörigen Übungen und Tutorien), die die zweite Vorlesung des dreisemestrigen Zyklus „Algorithmische Diskrete Mathematik“ bildet. Diese Vorlesung wurde von mir im Sommersemester 2015 zusammen mit Dr. Axel Werner an der TU Berlin gehalten, der auch an der Ausarbeitung des vorliegenden Vorlesungsskripts beteiligt war.

Wir gehen in dieser Vorlesung davon aus, dass die Studierenden die Vorlesung „Algorithmische Mathematik I (Einführung in die Lineare und Kombinatorische Optimierung)“ gehört haben und das zugehörige Vorlesungsskriptum

`www.zib.de/groetschel/teaching/WS1415/Skriptum_ADM_I-2015-02-16.pdf`

kennen. Wir werden im vorliegenden Skript häufig auf das ADM I Skript Bezug nehmen und übernehmen die Notation aus diesem Skript.

Der Inhalt dieser Vorlesung besteht aus einer (bunten) Mischung von Polyedertheorie, Matroid- und Graphentheorie, linearer, kombinatorischer und gemischt-ganzzahliger Optimierung. Einige Themen aus ADM I werden erneut aufgegriffen und vertieft. Wir versuchen dabei, verschiedene Aspekte der diskreten Mathematik miteinander zu verschränken und Bezüge zwischen den einzelnen Themenbereichen sichtbar zu machen. Die (algorithmische) diskrete Mathematik besteht nicht aus voneinander unabhängigen Einzelthemen und hochspezialisierten Werkzeugkästen, erst die Kombination der vielen Strukturen, Analysemethoden und algorithmischen Ansätzen macht diese sich stark entwickelnde Teildisziplin der Mathematik zu einem Gebiet, das neben schöner Theorie eine große Vielfalt an realen Anwendungen bietet.

Es gibt kein einzelnes Buch, das den gesamten, in dieser Vorlesung abgehandelten Themenkreis abdeckt. Daher sind in die einzelnen Kapitel Literaturhinweise eingearbeitet worden. Hinweise auf aktuelle Lehrbücher, die als Begleittexte zur Vorlesung geeignet sind finden sich auf der zur Vorlesung gehörigen Webseite:

`www.zib.de/groetschel/teaching/SS2015/Lecture-SS2015deutsch.html`

Die vorliegende Ausarbeitung ist ein Vorlesungsskript und kein Buch. Obwohl mit der gebotenen Sorgfalt geschrieben, war nicht genügend Zeit für das bei Lehrbüchern notwendige intensive Korrekturlesen und das Einarbeiten umfassender Literaturhinweise. Die daher vermutlich vorhandenen Fehler bitte ich zu entschuldigen (und mir wenn möglich mitzuteilen). Das Thema wird nicht erschöpfend behandelt. Das Manuskript enthält nur die wesentlichen Teile der Vorlesung. Insbesondere sind die in der Vorlesung erfolgten Schilderungen komplexer Anwendungsfälle, der Schwierigkeiten bei der mathematischen Modellierung praktischer Probleme, des Aufwandes, der bei der Implementierung der vorgestellten Algorithmen betrieben werden muss, der Probleme bei der praktischen Umsetzung und die Darstellung der Erfolge, die in den letzten Jahren beim Einsatz der hier vorgestellten Methodik in der Industrie erzielt wurden, nicht in das Skript aufgenommen worden.

Martin Grötschel

Literaturverzeichnis

R. K. Ahuja, T. L. Magnanti, und J. B. Orlin. *Network Flows, Theory, Algorithms and Applications.* Paramount Publishing International, Prentice Hall, New York, 1993.

D. Bertsimas und R. Weismantel. *Optimization over Integers.* Dynamic Ideas, Belmont, Massachusetts, 2005.

R. E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50:3–15, 2002.

M. Conforti, G. Cornuejols, und G. Zambelli. *Integer Programming.* Springer, 2014.

G. B. Dantzig. *Lineare Programmierung und Erweiterungen.* Springer, 1966.

R. Diestel. *Graph Theory.* Springer, 4th edition, 2010. `http://diestel-graph-theory.com`.

R. Fourer. Linear programming – software survey. *ORMS Today*, 42(3):52–63, 2015.

P. Gritzmann. *Grundlagen der Mathematischen Optimierung.* Springer, 2013.

M. Grötschel, L. Lovász, und A. Schrijver. *Geometric algorithms and combinatorial optimization.* Springer, 1993.

D. Jungnickel. *Graphs, Networks and Algorithms.* Springer, 4th edition, 2013.

B. Korte und J. Vygen. *Combinatorial Optimization – Theory and Algorithms.* Springer, Fifth edition, 2012.

S. O. Krumke und H. Noltemeier. *Graphentheoretische Konzepte und Algorithmen.* Teubner, Wiesbaden, 2005.

J. Matousek und B. Gärtner. *Using and Understanding Linear Programming.* Springer, 2006.

G. L. Nemhauser und L. A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, 1999.

M. Padberg. *Linear Optimization and Extensions.* Springer, 1995.

C. H. Papadimitriou und K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications Inc., 1998.

A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1998.

R. J. Vanderbei. *Linear Programming.* Springer, 2001.

Inhaltsverzeichnis

1 Polyedertheorie

Polyeder spielen in der Optimierung und in der Diskreten Mathematik eine große Rolle. Wir haben in ADM I in Abschnitt 2.3 wichtige Begriffe und Bezeichnungsweisen eingeführt und in Kapitel 8 einige grundlegende Resultate bewiesen. Wer sich nicht mehr an die Begriffe Kegel, Fourier-Motzkin-Elimination, Projektion, Seitenfläche, Facette, Ecke, spitz, oder gar den Dualitätssatz der linearen Optimierung erinnert, möge bitte im Manuskript der vorangegangenen Vorlesung ADM I nachschlagen.

Der zweite Teil des Vorlesungszyklus beginnt mit dem Ausbau der Polyedertheorie. Wir werden weitere Darstellungen von Polyedern angeben und neue Operationen mit Polyedern einführen, welche später an verschiedenen Stellen benötigt werden. Wir werden dabei meistens von der geometrischen Anschauung ausgehen und die Begriffe von dieser Sicht aus motivieren. Besonderes Augenmerk wird allerdings auch auf die analytische Beschreibung der geometrischen Konzepte gelegt. Weiterführende Details zur Polyedertheorie finden sich z. B. in Grünbaum (2003), Ziegler (2010) oder auch Matoušek (2002).

Wir erinnern daran, dass ein Polyeder P die Lösungsmenge eines linearen Ungleichungssystems $Ax \leq b$ ist und benutzen dafür die Bezeichnung $P = P(A, b)$. Polyeder der Form $\{x \in \mathbb{K}^n \mid Ax = b, x \geq 0\}$ bezeichnen wir mit $P^{=}(A, b)$. Ein Polytop ist ein beschränktes Polyeder.

1.1 Transformationen von Polyedern

Wir haben in der Vorlesung ADM I bereits Projektionen von Polyedern (entlang eines Richtungsvektors c) untersucht und in Satz (10.13) festgestellt, dass eine derartige Projektion wieder ein Polyeder ist. Wenn man mehrfach hintereinander projiziert, bleibt diese Eigenschaft natürlich erhalten. Sind $A \in \mathbb{K}^{(m,n)}$, $b \in \mathbb{K}^m$ und $k, r \geq 0$ mit $k + r = n$, so nennt man die Menge

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

eine *Projektion von $P(A, b)$ auf $\mathbb{K}^k$*. Hierbei ist $\tilde{A} \in \mathbb{K}^{(m,n)}$ eine Matrix, die durch Spaltenvertauschung aus A hervorgeht. Offensichtlich folgt aus Satz (10.13):

(1.1) Bemerkung. Jede Projektion eines Polyeders $P(A, b) \subseteq \mathbb{K}^n$ auf $\mathbb{K}^k$, $k \leq n$, ist ein Polyeder. △

Dieser Sachverhalt ist in größerer Allgemeinheit gültig. Erinnern wir uns daran, dass jede *affine Abbildung* $f : \mathbb{K}^n \to \mathbb{K}^k$ gegeben ist durch eine Matrix $D \in \mathbb{K}^{(k,n)}$ und einen

Vektor $d \in \mathbb{K}^k$, so dass

$$f(x) = Dx + d \quad \forall\, x \in \mathbb{K}^n.$$

Für derartige Abbildungen gilt:

(1.2) Satz. *Affine Bilder von Polyedern sind Polyeder.* △

Beweis. Seien $P = P(A, b) \subseteq \mathbb{K}^n$ ein Polyeder und $f(x) = Dx + d$ eine affine Abbildung von $\mathbb{K}^n$ in den $\mathbb{K}^k$, dann gilt

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

wobei

$$B := \begin{pmatrix} A & 0 \\ D & -I \\ -D & I \end{pmatrix}, \quad \overline{b} := \begin{pmatrix} b \\ -d \\ d \end{pmatrix}.$$

Wenden wir nun das Projektionsverfahren (10.11) aus ADM I iterativ auf $B, \overline{b}$ und die Richtungsvektoren $e_1, e_2, \ldots, e_n$ an, so erhalten wir nach Satz (10.12) ein System $C\binom{x}{y} \leq c$ mit $C = (0, \overline{C})$, und es gilt:

$$\begin{aligned} \forall\, y \in \mathbb{K}^k : \big(\exists\, x \in \mathbb{K}^n \text{ mit } B\begin{pmatrix} x \\ y \end{pmatrix} \leq \overline{b} &\iff \exists\, x \in \mathbb{K}^n \text{ mit } C\begin{pmatrix} x \\ y \end{pmatrix} \leq c \\ &\iff \overline{C}y \leq c\big). \end{aligned}$$

Daraus folgt $f(P) = \{y \in \mathbb{K}^k \mid \overline{C}y \leq c\}$ ist ein Polyeder. □

Man beachte, dass der Beweis von Satz (1.2) durch die Anwendung der Fourier-Motzkin-Elimination sogar ein Verfahren zur expliziten Konstruktion des affinen Bildes von $P(A, b)$ beinhaltet.

Wir erinnern hier an unsere Konventionen zur Bildung von linearen, affinen, konvexen und konischen Hüllen von Mengen und Matrizen, die in Abschnitt 2.2.3 des ADM I Skripts zu finden sind. Aus Satz (1.2) ergibt sich dann direkt die folgende (auch aus anderen Gründen unmittelbar einsichtige) Beobachtung.

(1.3) Korollar (Satz von Weyl). *Für jede Matrix $A \in \mathbb{K}^{(m,n)}$ gilt:*

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

△

Beweis. Wir zeigen die Behauptung für die konische Hülle. Alle anderen Fälle beweist man analog.

$$\begin{aligned}\operatorname{cone}(A) &= \{x \in \mathbb{K}^m \mid \exists\, y \geq 0 \text{ mit } x = Ay\} \\ &= \{x \in \mathbb{K}^m \mid \exists\, y \in \mathbb{K}^n \text{ mit } \begin{pmatrix} I & -A \\ -I & A \\ 0 & -I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq 0\}.\end{aligned}$$

Die letzte Menge ist die Projektion eines Polyeders im $\mathbb{K}^{m+n}$ auf den $\mathbb{K}^m$, also nach (1.1) bzw. (1.2) ein Polyeder. □

Offenbar besagt die obige Folgerung nichts anderes als: Die lineare, affine, konvexe oder konische Hülle einer endlichen Teilmenge des $\mathbb{K}^n$ ist ein Polyeder. Für die konische Hülle hat dies Weyl (1935) gezeigt (daher der Name für Korollar (1.3)).

(1.4) Korollar. *Die Summe $P = P_1 + P_2$ zweier Polyeder P_1, P_2 ist ein Polyeder.* △

Beweis. Es seien $P_1 = P(A, a)$, $P_2 = P(B, b)$, dann gilt:

$$\begin{aligned}P = P_1 + P_2 &= \{x + y \in \mathbb{K}^n \mid Ax \leq a, By \leq b\} \\ &= \{z \in \mathbb{K}^n \mid \exists\, x, y \in \mathbb{K}^n \text{ mit } Ax \leq a, By \leq b, z = x + y\} \\ &= \{z \in \mathbb{K}^n \mid \exists\, x, y \in \mathbb{K}^n \text{ mit } A(z - y) \leq a, B(z - x) \leq b, z = x + y\} \\ &= \{z \in \mathbb{K}^n \mid \exists\, x, y \in \mathbb{K}^n \text{ mit } D \begin{pmatrix} x \\ y \\ z \end{pmatrix} \leq \begin{pmatrix} a \\ b \\ 0 \end{pmatrix}\}\end{aligned}$$

mit

$$D = \begin{pmatrix} 0 & -A & A \\ -B & 0 & B \\ 1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix}.$$

Also ist P die Projektion eines Polyeders des $\mathbb{K}^{3n}$ auf den $\mathbb{K}^n$, und somit nach (1.1) ein Polyeder. □

Verbinden wir nun die Erkenntnis aus (1.3), dass $\operatorname{conv}(A)$ und $\operatorname{cone}(B)$ Polyeder sind, mit (1.4), so erhalten wir:

(1.5) Korollar. *Es seien $A \in \mathbb{K}^{(m,n)}$, $B \in \mathbb{K}^{(m,n')}$, dann gilt*

$$P = \operatorname{conv}(A) + \operatorname{cone}(B)$$

ist ein Polyeder. △

Die obige Folgerung erscheint (durch geschickte Vorbereitung) völlig trivial, sie ist jedoch eine durchaus beachtenswerte Erkenntnis, denn wir werden bald zeigen, dass in der Tat alle Polyeder von der Form $\operatorname{conv}(A) + \operatorname{cone}(B)$ sind.

1.2 Kegelpolarität

Es gibt mehrere Möglichkeiten die Umkehrung von (1.5) zu beweisen. Eine besonders elegante, die eine geometrische Version des Farkas-Lemmas benutzt, führt über die Kegelpolarität. Diese Operation mag zunächst nur als technisches Hilfsmittel erscheinen. Sie und ihre Verallgemeinerungen (allgemeine Polaritäten, Blocker, Antiblocker) sind jedoch bedeutende Methoden in der Polyedertheorie und der linearen sowie ganzzahligen Optimierung.

Wir beginnen mit einer Neuinterpretation des Farkas-Lemmas (11.2)(c) aus ADM I. Dieses besagt

$$\exists\, x \geq 0, Ax = b \iff \forall\, u\ (A^T u \geq 0 \Rightarrow u^T b \geq 0).$$

Durch diese Aussage sind offenbar auch alle rechten Seiten b charakterisiert, für die $x \geq 0$, $Ax = b$ eine Lösung hat. Nach Definition gilt $\text{cone}(A) = \{b \in \mathbb{K}^m \mid \exists\, x \geq 0 \text{ mit } Ax = b\}$, also können wir aus der Aussage (11.2)(c) des ADM I Skripts folgern:

(1.6) Bemerkung. Für alle Matrizen $A \in \mathbb{K}^{(m,n)}$ gilt:

$$\text{cone}(A) = \{b \in \mathbb{K}^m \mid u^T b \leq 0\ \forall u \in P(A^T, 0)\}. \qquad \triangle$$

Bemerkung (1.6) kann man geometrisch wie folgt beschreiben. Die Menge der zulässigen rechten Seiten b von $Ax = b$, $x \geq 0$ ist genau die Menge aller Vektoren $b \in \mathbb{K}^m$, welche einen stumpfen Winkel mit allen Vektoren des Kegels $P(A^T, 0)$ bilden. Allgemeiner definieren wir nun für jede beliebige Menge $S \subseteq \mathbb{K}^n$

$$S^\circ := \{y \in \mathbb{K}^n \mid y^T x \leq 0\ \forall x \in S\}.$$

S° ist die Menge aller Vektoren, die einen stumpfen Winkel mit allen Vektoren aus S bilden. S° heißt *polarer Kegel* von S. (Überzeugen Sie sich, dass S° ein Kegel ist!) Wir erinnern hier an das in der linearen Algebra definierte *orthogonale Komplement*

$$S^\perp := \{y \in \mathbb{K}^n \mid y^T x = 0\ \forall x \in S\}.$$

Offensichtlich gilt $S^\perp \subseteq S^\circ$. Unter Benutzung der obigen Definition können wir Bemerkung (1.6) nun auch wie folgt aufschreiben.

(1.7) Korollar. *Für alle Matrizen $A \in \mathbb{K}^{(m,n)}$ gilt*

$$P(A^T, 0)^\circ = \text{cone}(A) \quad \textit{und} \quad P(A, 0)^\circ = \text{cone}(A^T). \qquad \triangle$$

Korollar (1.7) und die vorher gemachten Beobachtungen wollen wir an einem Beispiel erläutern. Es sei

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

dann sind die Kegel $P(A, 0)$ und $P(A, 0)^\circ$ in Abbildung 1.1 gezeichnet.

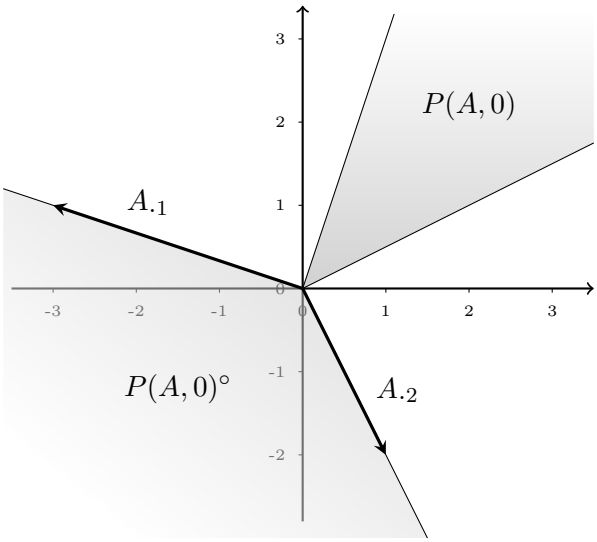


Abbildung 1.1: Kegel und polarer Kegel

$P(A,0)^\circ$ besteht also aus allen Vektoren, die mit den Elementen des Kegels $P(A,0)$ einen stumpfen Winkel bilden, und das sind gerade diejenigen Vektoren, die als konische Kombination der Normalenvektoren $A_{i\cdot}$ dargestellt werden können, also

$$P(A,0)^\circ = \operatorname{cone}\left(\begin{pmatrix}-3\\1\end{pmatrix}, \begin{pmatrix}1\\-2\end{pmatrix}\right).$$

Ferner gilt: $Ax = b$, $x \geq 0$ ist genau dann lösbar, wenn $b \in P(A,0)^\circ$ gilt. Daraus folgt z. B., dass $Ax = \binom{0}{1}$, $x \geq 0$ nicht lösbar ist, während $Ax = \binom{-1}{0}$, $x \geq 0$ eine Lösung hat.

Aus der Definition des polaren Kegels und des orthogonalen Komplements ergeben sich unmittelbar einige triviale Beziehungen, deren Beweis wir dem Leser zur Übung überlassen. Wir schreiben im Weiteren

$$S^{\circ\circ} := (S^\circ)^\circ.$$

(1.8) Bemerkung (Hausaufgabe). Für $S, S_i \subseteq \mathbb{K}^n$, $i = 1, \ldots, k$ gilt:

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

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

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

(d) $S^\circ = \operatorname{cone}(S^\circ) = (\operatorname{cone}(S))^\circ$

(e) $S = \operatorname{lin}(S) \Longrightarrow S^\circ = S^\perp$. Gilt die Umkehrung?

(f) Ersetzen wir in (a), …, (d) "$\circ$" durch "$\perp$", sind dann auch noch alle Behauptungen wahr? △

(1.9) Bemerkung (Hausaufgabe). Für welche Mengen $S \subseteq \mathbb{K}^n$ gilt

(a) $S^\circ = S^{\circ\circ\circ}$,

(b) $S = S^\circ$? △

Die Aussage (1.8)(d) impliziert insbesondere:

(1.10) Korollar. $\operatorname{cone}(A^T)^\circ = P(A, 0)$. △

Beweis. $(\operatorname{cone}(A^T))^\circ = \operatorname{cone}((A^T)^\circ) = (A^T)^\circ = \{x \mid Ax \leq 0\} = P(A, 0)$. □

Das folgende Korollar aus (1.7) und (1.10) wird in der Literatur häufig mit einem Namen belegt.

(1.11) Satz (Polarensatz). *Für jede Matrix $A \in \mathbb{K}^{(m,n)}$ gilt:*

$$P(A, 0)^{\circ\circ} = P(A, 0),$$
$$\operatorname{cone}(A)^{\circ\circ} = \operatorname{cone}(A).$$

△

Beweis.

$$P(A, 0) \overset{(1.10)}{=} \operatorname{cone}(A^T)^\circ \overset{(1.7)}{=} P(A, 0)^{\circ\circ},$$
$$\operatorname{cone}(A) \overset{(1.7)}{=} P(A^T, 0)^\circ \overset{(1.10)}{=} \operatorname{cone}(A)^{\circ\circ}.$$ □

Unser kurzer Exkurs über Kegelpolarität ist damit beendet.

1.3 Darstellungssätze

Wir wollen nun zeigen, dass Polyeder nicht nur in der Form $P(A, b)$ dargestellt werden können und benutzen dazu die bisher entwickelte Maschinerie.

(1.12) Satz (Minkowski (1896)). *Eine Teilmenge $K \subseteq \mathbb{K}^n$ ist genau dann ein polyedrischer Kegel, wenn K die konische Hülle von endlich vielen Vektoren ist. Mit anderen Worten: Zu jeder Matrix $A \in \mathbb{K}^{(m,n)}$ gibt es eine Matrix $B \in \mathbb{K}^{(n,k)}$, so dass*

$$P(A, 0) = \operatorname{cone}(B)$$

gilt und umgekehrt. △

Beweis. $P(A, 0) \overset{(1.10)}{=} \operatorname{cone}(A^T)^\circ \overset{(1.3)}{=} P(B^T, 0)^\circ \overset{(1.7)}{=} \operatorname{cone}(B)$. □

(1.13) Satz. *Es seien $A \in \mathbb{K}^{(m,n)}$, $b \in \mathbb{K}^m$, dann existieren endliche Mengen $V, E \subseteq \mathbb{K}^n$ mit*

$$P(A, b) = \operatorname{conv}(V) + \operatorname{cone}(E).$$ △

Beweis. Setze

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

dann gilt: $x \in P(A,b) \iff \binom{x}{1} \in H$. H ist nach Definition ein polyedrischer Kegel. Also gibt es nach Satz (1.12) eine Matrix $B \in \mathbb{K}^{(n+1,k)}$ mit $H = \text{cone}(B)$. Aufgrund der Definition von H hat die letzte Zeile von B nur nichtnegative Elemente. Durch Skalieren der Spalten von B und Vertauschen von Spalten können wir B in eine Matrix $\overline{B}$ so umformen, dass gilt

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

Daraus folgt:

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

□

(1.14) Korollar. *Eine Teilmenge $P \subseteq \mathbb{K}^n$ ist genau dann ein Polytop, wenn P die konvexe Hülle endlich vieler Vektoren ist.* △

Beweis. Sei $V \subseteq \mathbb{K}^n$ endlich und $P = \text{conv}(V)$, dann ist P nach (1.3) ein Polyeder. Ist $x \in P$, so gilt $x = \sum_{i=1}^k \lambda_i v_i$, $v_i \in V$, $\lambda_i \geq 0$, $\sum_{i=1}^k \lambda_i = 1$, und somit $\|x\| \leq \sum_{i=1}^k \|v_i\|$, d. h. $P \subseteq \{x \mid \|x\| \leq \sum_{v \in V} \|v\|\}$. Also ist P beschränkt, d. h. P ist ein Polytop.

Ist umgekehrt P ein Polytop, so gibt es nach Satz (1.13) endliche Mengen V, E mit $P = \text{conv}(V) + \text{cone}(E)$. Gibt es einen Vektor $e \in E$ mit $e \neq 0$, so gilt für alle $n \in \mathbb{N}$: $x + ne \in P$ für alle $x \in \text{conv}(V)$. Also ist P unbeschränkt, falls $E \setminus \{0\} \neq \emptyset$. Daraus folgt $E \in \{\emptyset, \{0\}\}$, und dann gilt trivialerweise $\text{conv}(V) = \text{conv}(V) + \text{cone}(E) = P$. □

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

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

Beweis. Kombiniere (1.12), (1.13), (1.14) und (1.5). □

Ist $P \subseteq \mathbb{K}^n$ ein Polyeder, so wissen wir nunmehr, dass es für P zwei mögliche Darstellungen gibt. Es gilt nämlich

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

wobei A eine (m,n)-Matrix, $b \in \mathbb{K}^m$ und V, E endliche Mengen sind. Diese beiden Darstellungen sind grundsätzlich verschieden, was natürlich in vielerlei Hinsicht nützlich sein

kann. Manche Aussagen über Polyeder sind völlig trivial, wenn man von der einen Beschreibung ausgeht, während sie aus der anderen Beschreibung nicht unmittelbar folgen.

Die Darstellung $P(A, b)$ nennt man auch *äußere Beschreibung* von P. Der Grund für diese Bezeichnung liegt darin, dass man das Polyeder P wegen

$$P \;=\; \bigcap_{i=1}^{m} \{x \mid A_i.x \leq b_i\} \;\subseteq\; \{x \mid A_i.x \leq b_i\},$$

als Durchschnitt von größeren Mengen betrachten kann. P wird sozusagen „von außen“ durch sukzessives Hinzufügen von Ungleichungen (bzw. Halbräumen) konstruiert.

Hingegen nennt man $\operatorname{conv}(V) + \operatorname{cone}(E)$ eine *innere Beschreibung* von P. Ist $E = \emptyset$, so ist die Bezeichnung offensichtlich, denn $V \subseteq P$ und somit wird P durch konvexe Hüllenbildung von Elementen von sich selbst erzeugt. Analoges gilt, wenn P ein polyedrischer Kegel ist. Sind jedoch V und E nicht leer, dann ist E nicht notwendigerweise eine Teilmenge von P, jedoch gelingt es eben aus den Vektoren $v \in V$ zusammen mit den Vektoren $e \in E$ das Polyeder P „von innen her“ zu konstruieren.

Die Sätze (1.12), (1.14) und (1.15) beinhalten weitere wichtige Charakterisierungen von polyedrischen Kegeln, Polytopen und Polyedern. Wir erinnern uns aus der linearen Algebra daran, dass jeder lineare Teilraum L des $\mathbb{K}^n$ eine endliche Basis hat, d. h. eine endliche Teilmenge B besitzt, so dass B linear unabhängig ist und $L = \operatorname{lin}(B)$ gilt. Die linearen Teilräume des $\mathbb{K}^n$ sind also diejenigen Teilmengen des $\mathbb{K}^n$, deren Elemente durch Linearkombinationen einer endlichen Menge erzeugt werden können. Nach (1.14) sind Polytope genau diejenigen Teilmengen des $\mathbb{K}^n$, die durch Konvexkombinationen einer endlichen Menge erzeugt werden können.

Wir werden später sehen, dass es sogar eine eindeutig bestimmte minimale (im Sinne der Mengeninklusion) endliche Menge $V \subseteq \mathbb{K}^n$ gibt mit $P = \operatorname{conv}(V)$, d. h. Polytope haben sogar eine eindeutig bestimmte „konvexe Basis“. Nach (1.12) sind polyedrische Kegel genau diejenigen Teilmengen des $\mathbb{K}^n$, die ein endliches „Kegelerzeugendensystem“ haben. Auch hier gibt es natürlich minimale endliche Mengen, die die Kegel konisch erzeugen. Aber nur unter zusätzlichen Voraussetzungen haben zwei minimale konische Erzeugendensysteme auch gleiche Kardinalität, und Eindeutigkeit gilt lediglich bis auf Multiplikation mit positiven Skalaren. Häufig nennt man eine Teilmenge T des $\mathbb{K}^n$ *endlich erzeugt*, falls $T = \operatorname{conv}(V) + \operatorname{cone}(E)$ für endliche Mengen V, E gilt. Nach (1.15) sind also die Polyeder gerade die endlich erzeugten Teilmengen des $\mathbb{K}^n$. Fassen wir zusammen, so gilt:

(1.16) Bemerkung. Ist $T \subseteq \mathbb{K}^n$, so gilt

(a) T ist ein linearer Teilraum $\iff$ T ist die lineare Hülle einer endlichen Menge.

(b) T ist ein affiner Teilraum $\iff$ T ist die affine Hülle einer endlichen Menge.

(c) T ist ein polyedrischer Kegel $\iff$ T ist die konische Hülle einer endlichen Menge.

(d) T ist ein Polytop $\iff$ T ist die konvexe Hülle einer endlichen Menge.

(e) T ist ein Polyeder $\iff$ T ist endlich erzeugt. △

Exkurs: Andere Darstellungsformen von Polyedern

Satz (1.15) und Bemerkung (1.16) zeigen, dass Polyeder auch durch Hüllenbildungsprozesse (linear, affin, konisch, konvex) und nicht nur durch Durchschnitte (Halbräume, Hyperebenen), die uns in ADM I (siehe (2.1)) zur Definition gedient haben, charakterisiert werden können. Dies sind jedoch nicht die einzigen Möglichkeiten, Polyeder zu beschreiben. Wir können hierauf nicht vertieft eingehen, sondern erwähnen nur zwei Beispiele.

Der harmlos aussehende absolute Betrag $|.|$ ermöglicht in manchen Fällen enorm kompakte Darstellungen. Wir betrachten als Beispiel

$$K(n) := \text{conv}\{e_1, \ldots, e_n, -e_1, \ldots, -e_n\},$$

wobei e_i den i-ten Einheitsvektor im $\mathbb{K}^n$ bezeichnet. $K(n)$ wird in der Literatur *Kreuzpolytop* genannt. Zur Definition des Kreuzpolytops $K(n)$ durch Hüllenbildung benötigt man also $2n$ Vektoren. Will man $K(n)$ als Durchschnitt von Halbräumen darstellen, so sind (beweisbar) 2^n Ungleichungen erforderlich:

$$K(n) = \{x \in \mathbb{K}^n \mid a^T x \leq 1 \; \forall a \in \{-1, 1\}^n\}.$$

Erlaubt man die Benutzung des Absolutbetrages, so ergibt sich

$$K(n) = \{x \in \mathbb{K}^n \mid \sum_{i=1}^{n} |x_i| \leq 1\}.$$

Eine einzige Ungleichung genügt in diesem Falle also zur Darstellung des Kreuzpolytops. Das Kreuzpolytop $K(3)$ im dreidimensionalen Raum ist das bekannte Oktaeder.

Tiefliegende Sätze der reellen algebraischen Geometrie, die auf Bröcker (1991) und Scheiderer (1989) zurückgehen, siehe hierzu Bochnak et al. (1998), zeigen, dass der *Stabilitätsindex* jeder „basic closed semi-algebraic set“ im Raum $\mathbb{R}^n$ den Wert $m := \frac{n(n+1)}{2}$ hat.

Polyeder sind spezielle „basic closed semi-algebraic sets“. Übersetzt in „unsere“ Sprache und bezogen auf Polyeder besagt das Resultat von Bröcker und Scheiderer, dass es zu jedem Polyeder P Polynome $p_1, \ldots, p_m$ in n reellen Variablen mit reellen Koeffizienten gibt, so dass

$$P = \{x \in \mathbb{R}^n \mid p_i(x) \geq 0, i = 1, \ldots, \frac{n(n+1)}{2}\}$$

gilt. Der Beweis ist rein „existenziell“ und liefert kein Konstruktionsverfahren für diese m Polynome. Es gibt allgemeine semi-algebraische Mengen, bei denen man auch beweisbar m Polynome braucht.

Für den Spezialfall von Polyedern wurde in Bosse et al. (2005) gezeigt, dass man im $\mathbb{R}^n$ die benötigten Polynome algorithmisch bestimmen kann und dass man sogar mit $2n$ Polynomen auskommt. Dieses Resultat wurde von Averkov und Bröcker (2012) verbessert. Sie zeigten, dass sogar n Polynome zur Darstellung von Polyedern ausreichen (und dass diese auch konstruiert werden können). Da es Polyeder gibt, für die man mindestens n Polynome zur Darstellung benötigt, ist dieses Resultat bestmöglich.

Eine Konsequenz der oben geschilderten Ergebnisse ist, dass das Kreuzpolytop $K(n)$ statt mit 2^n linearen Ungleichungen mit lediglich n Polynomungleichungen beschrieben werden kann.

1.4 Gültige Ungleichungen, Seitenflächen und Dimension

In Abschnitt 1.2 haben wir bereits einen Polarentyp, die Kegelpolare, zur Beweisvereinfachung eingeführt. Hier wollen wir eine weitere Polare betrachten, die es uns ermöglichen wird, eine Charakterisierung bezüglich $P(A, b)$ in eine Charakterisierung bezüglich $\text{conv}(V) + \text{cone}(E)$ zu übertragen.

(1.17) Definition. *Es seien $S \subseteq \mathbb{K}^n$, $a \in \mathbb{K}^n$, $\alpha \in \mathbb{K}$. Die Menge*

$$S^\gamma := \{\begin{pmatrix} a \\ \alpha \end{pmatrix} \in \mathbb{K}^{n+1} \mid a^T x \leq \alpha \; \forall x \in S\}$$

heißt γ-Polare von S. △

Die γ-Polare S^γ kann als die „Menge aller gültigen Ungleichungen bezüglich S“ betrachtet werden. Wir wollen nun die γ-Polare eines Polyeders charakterisieren.

(1.18) Satz. *Es sei $P \subseteq \mathbb{K}^n$, $P \neq \emptyset$, ein Polyeder mit den Darstellungen $P = P(A, b) = \text{conv}(V) + \text{cone}(E)$, dann gilt:*

(a) $P^\gamma = \left\{ \begin{pmatrix} a \\ \alpha \end{pmatrix} \in \mathbb{K}^{n+1} \mid \exists u \geq 0, u^T A = a^T, u^T b \leq \alpha \right\} = \text{cone} \begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix}.$

(b) $P^\gamma = \left\{ \begin{pmatrix} a \\ \alpha \end{pmatrix} \in \mathbb{K}^{n+1} \mid \begin{pmatrix} V^T & -\mathbb{1} \\ E^T & 0 \end{pmatrix} \cdot \begin{pmatrix} a \\ \alpha \end{pmatrix} \leq 0 \right\} = P\left(\begin{pmatrix} V^T & -\mathbb{1} \\ E^T & 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right).$ △

Beweis.

(a) $\begin{pmatrix} a \\ \alpha \end{pmatrix} \in P^\gamma \iff Ax \leq b, a^T x > \alpha$ inkonsistent

$\iff \exists u \geq 0, v > 0$ mit $u^T A - v a^T = 0, u^T b - v\alpha \leq 0$ (ADM I, (11.5))

$\iff \exists u \geq 0$ mit $u^T A = a^T, u^T b \leq \alpha$

$\iff \exists u \geq 0, \lambda \geq 0$ mit $\begin{pmatrix} a \\ \alpha \end{pmatrix} = \begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix} \begin{pmatrix} u \\ \lambda \end{pmatrix}$

$\iff \begin{pmatrix} a \\ \alpha \end{pmatrix} \in \text{cone} \begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix}.$

(b) $\begin{pmatrix} a \\ \alpha \end{pmatrix} \in P^\gamma \implies a^T v \leq \alpha \; \forall v \in V$ und $a^T(v + \lambda e) \leq \alpha \; \forall v \in V, e \in E, \lambda \geq 0$

$\implies a^T e \leq 0$ (andernfalls wäre $a^T(v + \lambda e) > \alpha$ für genügend großes λ)

$\implies \begin{pmatrix} V^T & -\mathbb{1} \\ E^T & 0 \end{pmatrix} \begin{pmatrix} a \\ \alpha \end{pmatrix} \leq 0.$

Gilt umgekehrt das letztere Ungleichungssystem, und ist $x \in P$, so existieren $v_1, \ldots, v_p \in V$ und $e_1, \ldots, e_q \in E$, $\lambda_1, \ldots, \lambda_p \geq 0$, $\sum_{i=1}^{p} \lambda_i = 1$, $\mu_1, \ldots, \mu_q \geq 0$, so dass

$$x = \sum_{i=1}^{p} \lambda_i v_i + \sum_{j=1}^{q} \mu_j e_j.$$

Und daraus folgt

$$a^T x \;=\; \sum_{i=1}^{p} \lambda_i a^T v_i + \sum_{j=1}^{q} \mu_j a^T e_j \;\leq\; \sum_{i=1}^{p} \lambda_i \alpha + \sum_{j=1}^{q} \mu_j 0 \;=\; \alpha,$$

also gilt $\begin{pmatrix} a \\ \alpha \end{pmatrix} \in P^\gamma$. □

(1.19) Korollar. *Die γ-Polare eines Polyeders $\emptyset \neq P \subseteq \mathbb{K}^n$ ist ein polyedrischer Kegel im $\mathbb{K}^{n+1}$.* △

(1.20) Korollar. *Ist $\emptyset \neq P = P(A, b) = \mathrm{conv}(V) + \mathrm{cone}(E)$ ein Polyeder und $a^T x \leq \alpha$ eine Ungleichung, dann sind äquivalent:*

(i) *$a^T x \leq \alpha$ ist gültig bezüglich P.*

(ii) *$\exists u \geq 0$ mit $u^T A = a^T$, $u^T b \leq \alpha$.*

(iii) *$a^T v \leq \alpha \; \forall v \in V$ und $a^T e \leq 0 \; \forall e \in E$.*

(iv) $\begin{pmatrix} a \\ \alpha \end{pmatrix} \in P^\gamma$. △

Wir erinnern an einige Begriffsbildungen aus der ADM I (siehe Definitionen (8.2) und (8.6) des ADM I Skripts): Eine Teilmenge $F \subseteq P$ eines Polyeders P ist eine *Seitenfläche von P*, wenn es eine gültige Ungleichung $c^T x \leq c_0$ bezüglich P gibt mit $F = P \cap \{x \mid c^T x = c_0\}$. Ist $P = P(A, b)$ und M die Zeilenindexmenge von A, dann ist für eine Teilmenge $F \subseteq P$ die *Gleichheitsmenge* $\mathrm{eq}(F) := \{i \in M \mid A_i.x = b_i \; \forall x \in F\}$, die Menge der für alle $x \in F$ bindenden Restriktionen. Für $I \subseteq M$ ist $\mathrm{fa}(I) := \{x \in P \mid A_I.x = b_I\}$ die *von I induzierte Seitenfläche.*

Wir zeigen nun, dass man die Gleichheitsmenge einer Seitenfläche explizit berechnen kann.

(1.21) Satz. *Seien $P = P(A, b)$ ein Polyeder und $\emptyset \neq F = \{x \in P \mid c^T x = c_0\}$ eine Seitenfläche von P. Dann gilt*

$$\mathrm{eq}(F) = \{i \in M \mid \exists u \geq 0 \text{ mit } u_i > 0 \text{ und } u^T A = c^T, u^T b = c_0\}.$$ △

Beweis. Nach Voraussetzung und Folgerung (11.19) aus ADM I haben die beiden linearen Programme

$$\begin{array}{l} \max\ c^T x \\ \quad Ax \le b \end{array} \quad \text{(P)} \qquad \text{und} \qquad \begin{array}{l} \min\ u^T b \\ \quad u^T A = c^T \\ \quad u \ge 0 \end{array} \quad \text{(D)}$$

optimale Lösungen mit gleichem Zielfunktionswert c_0, und F ist die Menge der Optimallösungen von (P). Sei nun $i \in \mathrm{eq}(F)$. Aufgrund des Satzes (11.26) vom starken komplementären Schlupf existieren Optimallösungen $\bar{x}$, $\bar{u}$ von (P), (D) mit $\bar{u}_j > 0 \Leftrightarrow A_{j.}\bar{x} = b_{j.}$. Wegen $\bar{x} \in F$ gilt $A_{i.}\bar{x} = b_i$, also gibt es einen Vektor u mit den geforderten Eigenschaften.

Gibt es umgekehrt einen Vektor $u \ge 0$ mit $u_i > 0$ und $u^T A = c^T$, $u^T b = c_0$, so ist u optimal für (D), und aus dem Satz vom schwachen komplementären Schlupf (11.25) folgt $A_{i.}x = b_i$ für alle $x \in F$, d. h. $i \in \mathrm{eq}(F)$. □

(1.22) Satz. *Seien $P = P(A, b)$ ein Polyeder und $F \neq \emptyset$ eine Teilmenge von P, dann sind äquivalent:*

(i) *F ist eine Seitenfläche von P.*

(ii) *$\exists I \subseteq M$ mit $F = \mathrm{fa}(I) = \{x \in P \mid A_{I.}x = b_I\}$.*

(iii) *$F = \mathrm{fa}(\mathrm{eq}(F))$.* △

Beweis. Gelten (ii) oder (iii), dann ist F offenbar eine Seitenfläche von P.

(i) $\Longrightarrow$ (ii): Sei $F = \{x \in P \mid c^T x = c_0\}$ eine Seitenfläche. Setzen wir $I := \mathrm{eq}(F)$, dann gilt nach Definition $F \subseteq \{x \in P \mid A_{I.}x = b_I\} =: F'$. Ist $x \in F'$, so gilt $x \in P$; es bleibt zu zeigen, dass $c^T x = c_0$ gilt. Zu jedem $i \in I$ gibt es nach (1.21) einen Vektor $u^{(i)}$ mit $u^{(i)} \ge 0$, $u_i^{(i)} > 0$, $(u^{(i)})^T A = c^T$ und $(u^{(i)})^T b = c_0$. Setze

$$u := \sum_{i \in I} \frac{1}{|I|} u^{(i)},$$

dann gilt nach Konstruktion $u_i > 0\ \forall i \in I$ und ferner $u_i = 0\ \forall i \in M \setminus I$ (andernfalls wäre $i \in I$ nach (1.21)). Die Vektoren x und u sind zulässig für die linearen Programme (P), (D) des Beweises von (1.21). Aus dem Satz vom schwachen komplementären Schlupf (11.25) aus ADM I folgt, dass sie auch optimal sind. Daraus folgt $c^T x = c_0$ und somit $x \in F$.

(ii) $\Longrightarrow$ (iii) folgt direkt aus dem obigen Beweis. □

Aus Satz (1.22) folgt, dass zur Darstellung einer Seitenfläche von $P(A, b)$ keine zusätzliche Ungleichung benötigt wird. Man braucht lediglich in einigen der Ungleichungen des Systems $Ax \le b$ Gleichheit zu fordern. Da jede nichtleere Seitenfläche auf diese Weise erzeugt werden kann, folgt:

(1.23) Korollar. *Sind $A \in \mathbb{K}^{(m,n)}$, $b \in \mathbb{K}^m$, dann hat das Polyeder $P(A, b)$ höchstens $2^m + 1$ Seitenflächen.* △

Beweis. $M = \{1, \ldots, m\}$ hat 2^m Teilmengen. Für jede Teilmenge $I \subseteq M$ ist $P \cap \{x \mid A_I.x = b_I\}$ eine Seitenfläche von P. Dazu kommt u. U. noch die leere Seitenfläche. □

Man kann Seitenflächen auf ähnliche Weise durch Einführung von Abbildungen analog zu eq bzw. fa bezüglich der Darstellung $P = \text{conv}(V) + \text{cone}(E)$ charakterisieren. Diese Kennzeichnungen von Seitenflächen sind jedoch technisch aufwendiger. Der interessierte Leser sei dazu auf Bachem und Grötschel (1982) verwiesen.

Wir wollen nun zeigen, dass man auch die Dimension einer Seitenfläche eines Polyeders explizit berechnen kann. Zunächst führen wir einen Hilfsbegriff ein.

(1.24) Definition. *Ein Element x eines Polyeders P heißt* innerer Punkt von P, *wenn x in keiner echten Seitenfläche von P enthalten ist.* △

Achtung! Innere Punkte eines Polyeders P sind nicht notwendig auch topologisch innere Punkte im Sinne der natürlichen Topologie des $\mathbb{K}^n$. Unsere inneren Punkte sind topologisch innere Punkte im Sinne der Relativtopologie auf P.

(1.25) Satz. *Jedes nichtleere Polyeder besitzt innere Punkte.* △

Beweis. Sei $P = P(A, b)$ und $I = \text{eq}(P(A, b))$, $J = M \setminus I$. Gilt $I = M$, so hat P keine echten Seitenflächen, also ist jedes Element von P ein innerer Punkt. Andernfalls ist das System $Ax \leq b$ äquivalent zu

$$A_I.x = b_I,$$
$$A_J.x \leq b_J.$$

P hat innere Punkte heißt dann, dass es ein x gibt mit $A_I.x = b_I$ und $A_J.x < b_J$. Zu jedem $i \in J$ existiert nach Definition ein Vektor $y^{(i)} \in P$ mit $A_i.y^{(i)} < b_i$. Setze $y := \frac{1}{|J|} \sum_{i \in J} y^{(i)}$, dann ist y Konvexkombination von Elementen von P, also $y \in P$, und es gilt $A_J.y < b_J$. Mithin ist y ein innerer Punkt von P. □

(1.26) Satz. *Sei F Seitenfläche eines Polyeders $P(A, b)$ und $\bar{x} \in F$. Der Vektor $\bar{x}$ ist ein innerer Punkt von F genau dann, wenn* $\text{eq}(\{\bar{x}\}) = \text{eq}(F)$. △

Beweis. $\bar{x}$ ist genau dann ein innerer Punkt von F, wenn die kleinste (im Sinne der Mengeninklusion) Seitenfläche von F, die $\bar{x}$ enthält, F selbst ist. Offenbar ist $\text{fa}(\text{eq}(\{\bar{x}\}))$ die minimale Seitenfläche von P, die $\bar{x}$ enthält. Daraus folgt die Behauptung. □

(1.27) Satz. *Ist $F \neq \emptyset$ eine Seitenfläche des Polyeders $P(A, b) \subseteq \mathbb{K}^n$, dann gilt*

$$\dim(F) = n - \text{rang}(A_{\text{eq}(F)}.).$$ △

Beweis. Sei $I := \text{eq}(F)$. Aus der linearen Algebra wissen wir, dass $n = \text{rang}(A_I.) + \dim(\text{kern}(A_I.))$ gilt. Zu zeigen ist also: $\dim(F) = \dim(\text{kern}(A_I.))$. Seien

$$r := \dim(\text{kern}(A_I.)) \quad \text{und} \quad s := \dim(F).$$

„$r \geq s$“: Da $\dim(F) = s$, gibt es $s+1$ affin unabhängige Vektoren $x_0, x_1, \ldots, x_s \in F$. Dann sind die Vektoren $x_1 - x_0, \ldots, x_s - x_0$ linear unabhängig und erfüllen $A_I.(x_i - x_0) = 0$. $\ker(A_I.)$ enthält also mindestens s linear unabhängige Vektoren, also gilt $r \geq s$.

„$s \geq r$“: Nach (1.25) besitzt F einen inneren Punkt $\bar{x} \in F$. Nach (1.26) gilt $\mathrm{eq}(\{\bar{x}\}) = \mathrm{eq}(F) = I$, und daraus folgt für $J := M \setminus I$:

$$A_I.\bar{x} = b_I,$$
$$A_J.\bar{x} < b_J.$$

Ist $r = 0$, so gilt $s \geq 0$ wegen $\bar{x} \in F$. Sei also $r \geq 1$, und $\{x_1, \ldots, x_r\}$ sei eine Basis von $\ker(A_I.)$. Für $p = 1, \ldots, r$ und $j \in J$ setze:

$$\delta_{jp} := \begin{cases} \infty, & \text{falls } A_j.x_p = 0 \\ \dfrac{b_j - A_j.\bar{x}}{A_j.x_p} & \text{andernfalls,} \end{cases}$$
$$\varepsilon := \min\{\delta_{jp} \mid j \in J, p \in \{1, \ldots, r\}\}.$$

(Setze $\varepsilon \neq 0$ beliebig, falls $\delta_{jp} = \infty$ für alle j, p.) Für $i \in I$ und alle $p \in \{1, \ldots, r\}$ gilt nun

$$A_i.(\bar{x} + \varepsilon x_p) \;=\; A_i.\bar{x} + \varepsilon A_i.x_p \;=\; A_i.\bar{x} \;=\; b_i,$$

da $A_i.x_p = 0$. Für $j \in J$ gilt

$$\begin{aligned} A_j.(\bar{x} + \varepsilon x_p) &= A_j.\bar{x} + \varepsilon A_j.x_p \\ &\leq A_j.\bar{x} + \delta_{jp} A_j.x_p \\ &= A_j.\bar{x} + b_j - A_j.\bar{x} \\ &= b_j. \end{aligned}$$

Daraus folgt $\bar{x} + \varepsilon x_p \in F$ für alle $p \in \{1, \ldots, r\}$. Da die Vektoren $\varepsilon x_1, \ldots, \varepsilon x_r$ linear unabhängig sind, sind die Vektoren $\bar{x}$, $\bar{x} + \varepsilon x_1, \ldots, \bar{x} + \varepsilon x_r$ affin unabhängig. Das heißt, F enthält mindestens $r + 1$ affin unabhängige Vektoren, und somit gilt $\dim(F) = s \geq r$. □

(1.28) Korollar. *$P = P(A, b) \subseteq \mathbb{K}^n$ sei ein nichtleeres Polyeder, dann gilt:*

(a) $\dim(P) = n - \mathrm{rang}(A_{\mathrm{eq}(P)}.)$.

(b) *Ist* $\mathrm{eq}(P) = \emptyset$*, dann ist P volldimensional (d. h.* $\dim(P) = n$*).*

(c) *Ist F eine echte Seitenfläche von P, dann gilt* $\dim(F) \leq \dim(P) - 1$. △

Mit Hilfe von Satz (1.27) kann man auch die affine Hülle einer Seitenfläche auf einfache Weise bestimmen.

(1.29) Satz. *Sei $F \neq \emptyset$ eine Seitenfläche des Polyeders $P(A, b)$, dann gilt*

$$\mathrm{aff}(F) = \{x \mid A_{\mathrm{eq}(F)}.x = b_{\mathrm{eq}(F)}\}.$$ △

Beweis. Es seien $I := \mathrm{eq}(F)$ und $T := \{x \mid A_I.x = b_I\}$. Offenbar ist T ein affiner Raum und wegen $F \subseteq T$ gilt $\mathrm{aff}(F) \subseteq \mathrm{aff}(T) = T$. Sei $s = \dim(F)$, dann folgt aus Satz (1.27), dass $\dim(\ker(A_I.)) = s$ und somit $\dim(T) = s$ gilt. Aus $\dim(\mathrm{aff}(F)) = \dim T$ und $\mathrm{aff}(F) \subseteq T$ folgt $\mathrm{aff}(F) = T$. □

1.5 Facetten und Redundanz

Wie wir bereits bemerkt haben, kann man zu einem Ungleichungssystem $Ax \leq b$ beliebig viele Ungleichungen hinzufügen, ohne die Lösungsmenge des Systems zu ändern. Wir wollen nun untersuchen, wie man ein gegebenes Polyeder mit möglichst wenigen Ungleichungen darstellen kann. Dies ist speziell für die lineare Optimierung wichtig, da der Rechenaufwand zur Auffindung einer Optimallösung in der Regel von der Anzahl der vorgelegten Ungleichungen abhängt. Gesucht wird also eine Minimaldarstellung eines Polyeders, um rechentechnische Vorteile zu haben. Es wird sich zeigen, dass hierbei diejenigen Ungleichungen, die maximale echte Seitenflächen eines Polyeders definieren, eine wesentliche Rolle spielen. Deshalb wollen wir derartige Seitenflächen untersuchen.

(1.30) Definition. *$Ax \leq b$ sei ein Ungleichungssystem, und M sei die Zeilenindexmenge von A.*

(a) *Sei $I \subseteq M$, dann heißt das System $A_I.x \leq b_I$* unwesentlich *oder* redundant bezüglich *$Ax \leq b$, wenn $P(A,b) = P(A_{M\setminus I}., b_{M\setminus I})$ gilt.*

(b) *Enthält $Ax \leq b$ ein unwesentliches Teilsystem $A_I.x \leq b_I$, dann heißt $Ax \leq b$* redundant*, andernfalls* irredundant*.*

(c) *Eine Ungleichung $A_i.x \leq b_i$ heißt* wesentlich *oder* nicht redundant bezüglich *$Ax \leq b$, wenn $P(A,b) \neq P(A_{M\setminus\{i\}}., b_{M\setminus\{i\}})$ gilt.*

(d) *Eine Ungleichung $A_i.x \leq b_i$ heißt* implizite Gleichung bezüglich *$Ax \leq b$, wenn $i \in \mathrm{eq}(P(A,b))$ gilt.*

(e) *Ein System $Ax \leq a$, $Bx = b$ heißt* irredundant*, wenn $Ax \leq a$ keine unwesentliche Ungleichung bezüglich des Systems $Ax \leq a$, $Bx \leq b$, $-Bx \leq -b$ enthält und B vollen Zeilenrang hat.*

(f) *Eine nichttriviale Seitenfläche F von $P(A,b)$ heißt* Facette von *$P(A,b)$, falls F in keiner anderen echten Seitenfläche von $P(A,b)$ enthalten ist.* △

Wir weisen ausdrücklich darauf hin, dass Redundanz bzw. Irredundanz keine Eigenschaft des Polyeders $P(A,b)$ ist, sondern eine Eigenschaft des Ungleichungssystems $Ax \leq b$. Wir werden sehen, dass ein Polyeder viele irredundante Beschreibungen haben kann. Ferner ist auch die Annahme falsch, dass man immer durch das gleichzeitige Weglassen aller unwesentlichen Ungleichungen eines Systems $Ax \leq b$ eine irredundante Beschreibung von $P(A,b)$ enthält. Wir wollen nun zunächst unwesentliche Ungleichungen charakterisieren.

(1.31) Satz. *Ein Ungleichungssystem $A_I.x \leq b_I$ ist unwesentlich bezüglich $Ax \leq b$ genau dann, wenn es eine Matrix $U \in \mathbb{K}_+^{(|I|,m)}$ gibt mit $UA = A_I.$, $Ub \leq b_I$ und $U_{.I} = 0$.* △

Beweis. Für jedes $i \in I$ ist nach (1.20) die Ungleichung $A_i.x \leq b_i$ gültig bezüglich $P(A_{M\setminus I}., b_{M\setminus I})$ genau dann, wenn es einen Vektor $\bar{u}_i \geq 0$ gibt mit $\bar{u}_i^T A_{M\setminus I}. = A_i.$ und

$\bar{u}_i^T b_{M\setminus I.} \leq b_i$. Dies ist genau dann der Fall, wenn es eine Matrix $U \in \mathbb{K}_+^{(|I|,m)}$ gibt mit $UA = A_{I.}$, $Ub \leq b_I$ und $U_{.I} = 0$. □

(1.32) Korollar. *$A_{i.}x \leq b_i$ ist genau dann redundant bezüglich $Ax \leq b$, wenn es einen Vektor $u \in \mathbb{K}_+^m$ gibt mit $u^T A = A_{i.}$, $u^T b \leq b_i$, $u_i = 0$.* △

Der nächste Satz zeigt, wann man Ungleichungen nicht mehr weglassen kann, ohne das Polyeder zu ändern.

(1.33) Satz. *$P = P(A,b) \neq \emptyset$ sei ein Polyeder. Sei $\emptyset \neq I \subseteq M \setminus \mathrm{eq}(P)$ und $P' := P(A_{M\setminus I.}, b_{M\setminus I})$. Dann gilt*

$$P \neq P' \iff \exists \text{ nichttriviale Seitenfläche } F \subseteq P \text{ mit } \mathrm{eq}(F) \subseteq I \cup \mathrm{eq}(P). \qquad \triangle$$

Beweis. Im Weiteren bezeichnen wir mit $\mathrm{eq}_{P'}$ die „equality set"-Abbildung bezüglich P'. Es gilt offenbar $\mathrm{eq}_{P'}(F) \subseteq \mathrm{eq}(F)$.

„$\Longleftarrow$" Angenommen, es gilt $P = P'$, und F sei eine beliebige nichttriviale Seitenfläche von P (und somit auch von P'). Da F eine nichttriviale Seitenfläche von P' ist, gibt es ein $i \in (M \setminus I) \setminus \mathrm{eq}_{P'}(P')$ mit $i \in \mathrm{eq}_{P'}(F)$. Daraus folgt $\mathrm{eq}(F) \not\subseteq I \cup \mathrm{eq}(P)$.

„$\Longrightarrow$" Angenommen, es gilt $P \neq P'$. Wegen $P \subseteq P'$ heißt dies, es existiert ein Vektor $v \in P' \setminus P$, und somit gibt es eine Indexmenge $\emptyset \neq K \subseteq I$ mit der Eigenschaft $A_{i.}v \leq b_i \; \forall i \in M \setminus K$ und $A_{i.}v > b_i \; \forall i \in K$. Nach Satz (1.25) hat P einen inneren Punkt, sagen wir w, d. h. es gilt $A_{i.}w = b_i \; \forall i \in \mathrm{eq}(P)$ und $A_{i.}w < b_i \; \forall i \in M \setminus \mathrm{eq}(P)$.

Wir betrachten nun einen Punkt y auf der Strecke zwischen v und w, d. h. $y = \lambda w + (1-\lambda)v$ mit $0 \leq \lambda \leq 1$. Aufgrund der Voraussetzungen gilt:

$$\begin{aligned} A_{i.}y &= b_i \quad \forall i \in \mathrm{eq}(P) \\ A_{i.}y &< b_i \quad \forall i \in M \setminus (\mathrm{eq}(P) \cup K), \text{ falls } \lambda > 0. \end{aligned}$$

Ist $i \in K$, so gilt

$$\begin{aligned} A_{i.}y \leq b_i &\iff \lambda A_{i.}w + (1-\lambda)A_{i.}v \leq b_i \\ &\iff \lambda A_{i.}(w-v) \leq b_i - A_{i.}v \\ &\iff \lambda \geq \frac{b_i - A_{i.}v}{A_{i.}(w-v)} \quad (\text{da } A_{i.}(w-v) < 0). \end{aligned}$$

Setzen wir

$$\mu := \max\left\{ \frac{b_i - A_{i.}v}{A_{i.}(w-v)} \mid i \in K \right\},$$
$$L := \left\{ i \in K \mid \mu = \frac{b_i - A_{i.}v}{A_{i.}(w-v)} \right\},$$

dann gilt $z := \mu w + (1-\mu)v \in P$, $\emptyset \neq L \subseteq K \subseteq I$ und

$$\begin{aligned} A_{i.}z &= b_i \; \forall i \in L \\ A_{i.}z &< b_i \; \forall i \in K \setminus L. \end{aligned}$$

Daraus folgt, z ist ein innerer Punkt von $F := \mathrm{fa}(L \cup \mathrm{eq}(P))$. Nach (1.26) gilt dann $\mathrm{eq}(F) = \mathrm{eq}(\{z\}) = L \cup \mathrm{eq}(P)$, und das bedeutet, dass F eine nichttriviale Seitenfläche von P mit $\mathrm{eq}(F) \subseteq I \cup \mathrm{eq}(P)$ ist. □

Wir werden nun wichtige Eigenschaften von Facetten bestimmen, Nichtredundanz kennzeichnen und Facetten charakterisieren.

(1.34) Satz. *Sei F eine Facette von $P = P(A, b)$, dann gilt:*

(a) $\mathrm{eq}(P) \subsetneqq \mathrm{eq}(F)$.

(b) *Für alle $i \in \mathrm{eq}(F) \setminus \mathrm{eq}(P)$ gilt*

$$F = \mathrm{fa}(\{i\}) = \{x \in P \mid A_i.x = b_i\}. \qquad \triangle$$

Beweis. (a) gilt offensichtlich für alle nichttrivialen Seitenflächen von P.
(b) Die Abbildung fa ist inklusionsumkehrend, d. h.

$$I \subseteq J \implies \mathrm{fa}(I) \supseteq \mathrm{fa}(J).$$

Daraus folgt $F = \mathrm{fa}(\mathrm{eq}(F)) \subseteq \mathrm{fa}(\{i\})$. Da $i \notin \mathrm{eq}(P)$, muss $\mathrm{fa}(\{i\})$ eine echte Seitenfläche von P sein. Aus der Maximalität von F folgt die Behauptung. □

(1.35) Korollar. *Sei $P = P(A, b)$ ein Polyeder und $\mathcal{F}$ die Menge der Facetten von P. Dann gilt:*

(a) $F_1, F_2 \in \mathcal{F},\ F_1 \neq F_2 \implies \mathrm{eq}(F_1) \cap \mathrm{eq}(F_2) = \mathrm{eq}(P)$.

(b) $|\mathcal{F}| \leq m - |\mathrm{eq}(P)|$.

(c) *Es gibt eine Menge $I \subseteq M$ mit folgenden Eigenschaften*
   (c_1) $I \subseteq M \setminus \mathrm{eq}(P)$,
   (c_2) $|I| = |\mathcal{F}|$,
   (c_3) $F \in \mathcal{F} \iff \exists$ *genau ein* $i \in I$ *mit* $F = \mathrm{fa}(\{i\})$. $\triangle$

Jede Menge $I \subseteq M$ mit den Eigenschaften (c_1), (c_2), (c_3) wollen wir *Facetten-Indexmenge* nennen. Satz (1.34)(b) zeigt, dass man Facetten von P dadurch erhält, dass man in nur einer Ungleichung $A_i.x \leq b_i$ des Systems $Ax \leq b$ Gleichheit fordert. Jedoch ist es keineswegs so, dass für alle $i \in M$ die Menge $\mathrm{fa}(\{i\})$ eine Facette von P ist! Dies gilt nur für solche $i \in M$, die in einer Facettenindexmenge enthalten sind.

(1.36) Satz. *Seien $P = P(A, b) \neq \emptyset$ ein Polyeder und $\mathcal{F}$ die Menge der Facetten von P. Seien M die Zeilenindexmenge von A, $I \subseteq M \setminus \mathrm{eq}(P)$ und $J \subseteq \mathrm{eq}(P)$. Sei $P' := \{x \mid A_J.x = b_J, A_I.x \leq b_I\}$, dann gilt:*

(a) $P = P' \iff$ (a_1) $\forall F \in \mathcal{F}$ *gilt* $I \cap \mathrm{eq}(F) \neq \emptyset$ *und*
   (a_2) $\mathrm{rang}(A_J.) = \mathrm{rang}(A_{\mathrm{eq}(P)}.)$.

(b) $P = P(A_{I\cup \mathrm{eq}(P)\cdot}, b_{I\cup \mathrm{eq}(P)}) \iff \forall F \in \mathcal{F}$ *gilt* $I \cap \mathrm{eq}(F) \neq \emptyset$. △

Beweis. Mit $J = \mathrm{eq}(P)$ folgt (b) direkt aus (a). Wir beweisen (a).

„$\Longrightarrow$“ Nach Definition gilt offenbar $J = \mathrm{eq}_{P'}(P')$. Angenommen (a_2) ist nicht erfüllt, d. h. $\mathrm{rang}(A_{J\cdot}) < \mathrm{rang}(A_{\mathrm{eq}(P)\cdot})$. Dann folgt aus der Dimensionsformel (1.28)(a) $\dim(P') > \dim(P)$ und somit muss $P \neq P'$ gelten. Widerspruch!

Angenommen (a_1) ist nicht erfüllt. Dann gibt es eine Facette F von P mit $\mathrm{eq}(F) \subseteq M \setminus I = (M \setminus I) \cup \mathrm{eq}(P)$. Folglich gilt $P \neq P'$ nach Satz (1.33). Widerspruch!

„$\Longleftarrow$“ Wir zeigen zunächst, dass unter der Voraussetzung (a_2) gilt:

$$A_{J\cdot}x = b_J \implies A_{\mathrm{eq}(P)\cdot}x = b_{\mathrm{eq}(P)}.$$

Da $P' \neq \emptyset$, gilt $\mathrm{rang}(A_{J\cdot}, b_J) = \mathrm{rang}(A_{J\cdot}) = \mathrm{rang}(A_{\mathrm{eq}(P)\cdot}) = \mathrm{rang}(A_{\mathrm{eq}(P)\cdot}, b_{\mathrm{eq}(P)})$. Das heißt, für alle $i \in \mathrm{eq}(P)$ existieren $K \subseteq J$ und λ_k, $k \in K$, mit $A_{i\cdot} = \sum_{k\in K} \lambda_k A_{k\cdot}$, $b_i = \sum_{k\in K} \lambda_k b_k$. Erfüllt also der Vektor x das System $A_{J\cdot}x = b_J$, so gilt für alle $i \in \mathrm{eq}(P)$

$$A_{i\cdot}x = \sum_{k\in K} \lambda_k A_{k\cdot}x = \sum_{k\in K} \lambda_k b_k = b_i.$$

Nach (a_1) gilt für jede Facette F von P: $\mathrm{eq}(F) \not\subseteq M \setminus I$, und da Facetten maximale echte Seitenflächen sind und eq inklusionsumkehrend ist, folgt daraus $\mathrm{eq}(G) \not\subseteq M \setminus I$ für alle echten Seitenflächen G von P. Aus Satz (1.33) folgt daher $P = P'$. □

(1.37) Korollar. *Seien* $P = P(A, b) \neq \emptyset$ *ein Polyeder,* $I \subseteq M \setminus \mathrm{eq}(P)$, $J \subseteq \mathrm{eq}(P)$ *und* $P = \{x \mid A_{J\cdot}x = b_J, A_{I\cdot}x \leq b_I\}$. *Diese Darstellung von* P *ist genau dann irredundant, wenn gilt:*

(a) I *ist eine Facetten-Indexmenge von* P.

(b) $A_{J\cdot}$ *ist eine* $(\mathrm{rang}(A_{\mathrm{eq}(P)\cdot}), n)$*-Matrix mit vollem Zeilenrang.* △

(1.38) Korollar. *Sei* $P = P(A, b) \subseteq \mathbb{K}^n$ *ein volldimensionales Polyeder (also* $\mathrm{eq}(P) = \emptyset$*, bzw.* $\dim(P) = n$*), dann gilt für alle* $I \subseteq M$

$P(A_{I\cdot}, b_I)$ *ist eine irredundante Beschreibung von* P
$\iff$ I *ist Facetten-Indexmenge von* P. △

(1.39) Satz. *Sei* $P = P(A, b)$ *ein Polyeder, und* F *sei eine nichttriviale Seitenfläche von* P. *Dann sind äquivalent:*

(i) F *ist eine Facette von* P.

(ii) F *ist eine maximale echte Seitenfläche von* P.

(iii) $\dim(F) = \dim(P) - 1$.

(iv) *F enthält* $\dim(P)$ *affin unabhängige Vektoren.*

(v) *Sei $c^T x \leq c_0$ eine bezüglich P gültige Ungleichung mit $F = \{x \in P \mid c^T x = c_0\}$, dann gilt für alle gültigen Ungleichungen $d^T x \leq \delta$ mit $F \subseteq \{x \in P \mid d^T x = \delta\}$: Es gibt einen Vektor $u \in \mathbb{K}^{\mathrm{eq}(P)}$ und $\alpha \in \mathbb{K}$, $\alpha \geq 0$ mit*

$$d^T = \alpha c^T + u^T A_{\mathrm{eq}(P)\cdot},$$
$$\delta = \alpha c_0 + u^T b_{\mathrm{eq}(P)}. \qquad \triangle$$

Beweis. (i) $\iff$ (ii): nach Definition.

(iv) $\iff$ (iii): trivial.

(iii) $\implies$ (ii): Angenommen F ist keine Facette, dann existiert eine echte Seitenfläche G von P mit $F \subset G \subset P$. Aus (1.28)(c) folgt dann $\dim(F) \leq \dim(G) - 1 \leq \dim(P) - 2$, Widerspruch!

(i) $\implies$ (v): Sei F eine beliebige Facette von P. Wir nehmen zunächst an, dass $A_I.x \leq b_I$, $A_J.x = b_J$ eine irredundante Darstellung von $P(A, b)$ ist mit $1 \in I$ und dass $F = \{x \in P \mid A_1.x = b_1\}$ gilt. Sei nun $d^T x \leq \delta$ eine gültige Ungleichung mit $F \subseteq \{x \in P \mid d^T x = \delta\}$. Aufgrund von Folgerung (1.20) gibt es Vektoren $v \geq 0$ und w mit $v^T A_{I\cdot} + w^T A_{J\cdot} = d^T$ und $v^T b_I + w^T b_J \leq \delta$ (in der Tat gilt hier Gleichheit, da $\{x \mid d^T x = \delta\}$ eine Stützhyperebene ist). Angenommen, es gibt einen Index $i \in I \setminus \{1\}$ mit $v_i > 0$, dann gilt nach (1.21) $i \in \mathrm{eq}(F)$. Dies ist aber ein Widerspruch dazu, dass I eine Facettenindexmenge ist. Hieraus folgt (v).

(v) $\implies$ (iii): Da F eine echte Seitenfläche von P ist, gilt $\dim(F) \leq \dim(P) - 1$. Angenommen $\dim(F) \leq \dim(P) - 2$. O. B. d. A. können wir annehmen, dass $F = \{x \in P \mid A_1.x = b_1\}$ gilt. Aus (1.27) folgt

$$\mathrm{rang}(A_{\mathrm{eq}(F)\cdot}) \geq \mathrm{rang}(A_{\mathrm{eq}(P)\cdot}) + 2.$$

Mithin gibt es einen Index $i \in \mathrm{eq}(F) \setminus (\mathrm{eq}(P) \cup \{1\})$, so dass der Zeilenvektor $A_i.$ linear unabhängig von den Zeilenvektoren $A_j.$, $j \in \mathrm{eq}(P) \cup \{1\}$, ist. Das aber heißt, dass das System

$$A_i. = \alpha A_1. + u^T A_{\mathrm{eq}(P)\cdot}$$

keine Lösung α, u hat. Wegen $F \subseteq \{x \in P \mid A_i.x = b_i\}$ ist dies ein Widerspruch zu (v). $\square$

(1.40) Korollar. *Seien $P = P(A, b) \subseteq \mathbb{K}^n$ ein volldimensionales Polyeder und $F = \{x \in P \mid c^T x = c_0\}$ eine Seitenfläche von P. Dann sind äquivalent:*

(i) *F ist Facette von P.*

(ii) $\dim(F) = n - 1$.

(iii) *Für alle gültigen Ungleichungen* $d^T x \leq \delta$, $d \neq 0$, *mit* $F \subseteq \{x \in P \mid d^T x = \delta\}$ *gilt: Es existiert ein* $\alpha > 0$ *mit*

$$d^T = \alpha c^T,$$
$$\delta = \alpha c_0.$$

△

(1.41) Beispiel. Wir betrachten das Polyeder $P = P(A, b) \subseteq \mathbb{R}^2$, das wie folgt gegeben ist (siehe Abbildung 1.2).

$$A = \begin{pmatrix} A_{1.} \\ A_{2.} \\ A_{3.} \\ A_{4.} \\ A_{5.} \\ A_{6.} \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -2 & 2 \\ 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ -1 & -1 \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 2 \\ -1 \\ -2 \end{pmatrix}.$$

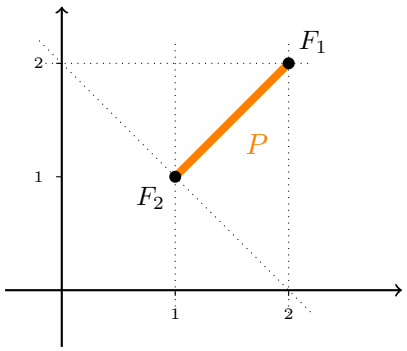


Abbildung 1.2: Ein 1-dimensionales Polyeder in $\mathbb{R}^2$

P hat 4 Seitenflächen, nämlich $\emptyset$, P und $F_1 = \{\binom{2}{2}\}$, $F_2 = \{\binom{1}{1}\}$. F_1 und F_2 sind Facetten von P. Es gilt $\text{eq}(P) = \{1, 2\}$, $\text{eq}(F_1) = \{1, 2, 3, 4\}$, $\text{eq}(F_2) = \{1, 2, 5, 6\}$, $\text{eq}(\emptyset) = \{1, \ldots, 6\}$. Die Mengen $\{3, 5\}$, $\{3, 6\}$, $\{4, 5\}$, $\{4, 6\}$ sind die Facettenindexmengen von P. Eine irredundante Beschreibung von P ist z. B. gegeben durch

$$P = \{x \mid A_{1.}x = 0, A_{3.}x \leq b_3, A_{5.}x \leq b_5\}.$$

Übrigens sind die Ungleichungen $A_{i.}x \leq b_i$, $i = 3, 4, 5, 6$ redundant bezüglich $P(A, b)$. Die Ungleichungssysteme $A_{I.}x \leq b_I$ mit $I = \{3, 5\}$ oder $I = \{4, 6\}$ sind z. B. ebenfalls redundant. Aber $A_{I.}x \leq b_I$ ist nicht redundant bezüglich $P(A, b)$, falls $I = \{3, 4, 5\}$. △

1.6 Rezessionskegel, Linienraum und Homogenisierung

An dieser Stelle ist es nützlich einige weitere Objekte einzuführen, die man Polyedern (bzw. allgemeinen Mengen) zuordnen kann. Das Studium dieser Objekte ist für sich

selbst betrachtet sehr interessant. Wir wollen diese Mengen jedoch nur als Hilfsmittel zur Vereinfachung von Beweisen verwenden, weswegen wir nicht weiter auf theoretische Untersuchungen dieser Mengen eingehen werden.

(1.42) Definition. *Sei $S \subseteq \mathbb{K}^n$ eine beliebige Menge. Wir definieren*

(a) $\operatorname{rec}(S) := \{y \in \mathbb{K}^n \mid \exists x \in S,$ *so dass* $\forall \lambda \geq 0$ *gilt* $x + \lambda y \in S\}$,

(b) $\operatorname{lineal}(S) := \{y \in \mathbb{K}^n \mid \exists x \in S,$ *so dass* $\forall \lambda \in \mathbb{K}$ *gilt* $x + \lambda y \in S\}$,

(c) $\operatorname{hog}(S) := \{\binom{x}{1} \in \mathbb{K}^{n+1} \mid x \in S\}^{\circ\circ}$.

Die Menge $\operatorname{rec}(S)$ *heißt* Rezessionskegel *von* S, $\operatorname{lineal}(S)$ *heißt* Linealitätsraum *oder* Linienraum von S, *und* $\operatorname{hog}(S)$ *heißt* Homogenisierung von S. △

Wir wollen nun die oben eingeführten Mengen bezüglich Polyedern charakterisieren. Nennen wir einen Vektor y mit $x + \lambda y \in S$ für alle $\lambda \geq 0$ eine „Richtung nach Unendlich", so besteht der Rezessionskegel einer Menge S aus allen Richtungen nach Unendlich. Für Polyeder gilt Folgendes:

(1.43) Satz. *Sei $P = P(A, b) = \operatorname{conv}(V) + \operatorname{cone}(E)$ ein nichtleeres Polyeder, dann gilt*

$$\operatorname{rec}(P) = P(A, 0) = \operatorname{cone}(E). \qquad \triangle$$

Beweis. (a) $\operatorname{rec}(P) = P(A, 0)$.

Ist $y \in \operatorname{rec}(P)$, so existiert ein $x \in P$ mit $x + \lambda y \in P$ $\forall \lambda \geq 0$. Daraus folgt $b \geq A(x + \lambda y) = Ax + \lambda Ay$. Gäbe es eine Komponente von Ay, die größer als Null ist, sagen wir $(Ay)_i > 0$, so wäre der Vektor $x + \lambda_0 y$ mit

$$\lambda_0 = \frac{b_i - (Ax)_i}{(Ay)_i} + 1$$

nicht in $P(A, b)$, Widerspruch!

Ist $y \in P(A, 0)$, so gilt für alle $x \in P(A, b)$ und $\lambda \geq 0$, $A(x+\lambda y) = Ax+\lambda Ay \leq b+0 = b$, also ist $y \in \operatorname{rec}(P)$.

(b) $\operatorname{rec}(P) = \operatorname{cone}(E)$.

Die Inklusion $\operatorname{cone}(E) \subseteq \{y \in \mathbb{K}^n \mid \forall x \in S, \forall \lambda \geq 0 : x + \lambda y \in S\} \subseteq \operatorname{rec}(P)$ ist offensichtlich. Umgekehrt sei $y \in \operatorname{rec}(P)$, dann existiert wieder ein $x \in P$ wie oben. Angenommen $y \notin \operatorname{cone}(E)$, dann gibt es nach dem Farkas-Lemma (ADM I Skript (11.2)(c)) ein u mit $u^T E \leq 0$ und $u^T y > 0$. Für jedes $z \in P$ folgt dann mit gewissen λ_i, $0 \leq \lambda_i \leq 1$, $\sum_i \lambda_i = 1$ und $\mu_i \geq 0$:

$$\begin{aligned} u^T z &= u^T \sum_i \lambda_i V_{\cdot i} + u^T \sum_i \mu_i E_{\cdot i} = \sum_i \lambda_i u^T V_{\cdot i} + u^T E \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_{|E|} \end{pmatrix} \leq \sum_i \lambda_i u^T V_{\cdot i} \\ &\leq \max_i u^T V_{\cdot i} \end{aligned}$$

Andererseits gilt aber $u^T(x + \lambda y) = u^T x + \lambda u^T y \to \infty$, für $\lambda \to \infty$, ein Widerspruch zu $x + \lambda y \in P$ für alle $\lambda \geq 0$. □

Insbesondere folgt aus dem Beweis auch, dass $\operatorname{rec}(P) = \{y \in \mathbb{K}^n \mid \forall x \in P \text{ und } \forall \lambda \geq 0 \text{ gilt } x + \lambda y \in P\}$ für Polyeder P gilt. Ist P ein Kegel, so gilt natürlich $P = \operatorname{rec}(P)$, und offenbar ist ein Polyeder P genau dann ein Polytop, wenn $\operatorname{rec}(P) = \{0\}$. Abbildung 1.3 zeigt ein Polyeder und seinen Rezessionskegel.

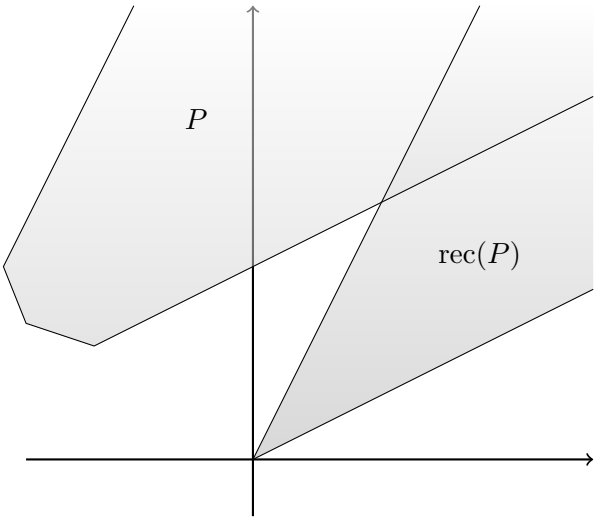


Abbildung 1.3: Ein Polyeder und sein Rezessionskegel

Aus Definition (1.42) folgt $\operatorname{lineal}(P) = \operatorname{rec}(P) \cap (-\operatorname{rec}(P))$. Offenbar ist $\operatorname{lineal}(P)$ ein linearer Teilraum des $\mathbb{K}^n$, und zwar ist es der größte lineare Teilraum $L \subseteq \mathbb{K}^n$, so dass $x + L \subseteq P$ für alle $x \in P$ gilt. Analytisch können wir $\operatorname{lineal}(P)$ wie folgt darstellen.

(1.44) Satz. *Sei $P = P(A, b) = \operatorname{conv}(V) + \operatorname{cone}(E)$ ein nichtleeres Polyeder, dann gilt*

$$\operatorname{lineal}(P) = \{x \mid Ax = 0\} = \operatorname{cone}(\{e \in E \mid -e \in \operatorname{cone}(E)\}). \qquad \triangle$$

Beweis. Wegen $\operatorname{lineal}(P) = \operatorname{rec}(P) \cap (-\operatorname{rec}(P))$ folgt die Behauptung direkt aus (1.43).□

Wir kommen nun zur Homogenisierung. Die Definition der Homogenisierung erscheint etwas kompliziert: Man wende zweimal die Kegelpolarität auf die Menge S an! Geometrisch betrachtet ist $\operatorname{hog}(S)$ der Durchschnitt aller Ungleichungen mit rechter Seite 0, die gültig bezüglich $\{\binom{x}{1} \mid x \in S\}$ sind.

(1.45) Satz. *Sei $P = P(A, b) = \operatorname{conv}(V) + \operatorname{cone}(E)$ ein nichtleeres Polyeder. Sei*

$$B = \begin{pmatrix} A & -b \\ 0 & -1 \end{pmatrix},$$

dann gilt

$$\operatorname{hog}(P) = P(B, 0) = \operatorname{cone}(\{\tbinom{v}{1} \mid v \in V\}) + \operatorname{cone}(\{\tbinom{e}{0} \mid e \in E\}). \qquad \triangle$$

Beweis. Setzen wir $P_1 := \{\binom{x}{1} \in \mathbb{K}^{n+1} \mid x \in P\}$, so gilt offensichtlich

$$P_1 = \operatorname{conv}(\{\tbinom{v}{1} \mid v \in V\}) + \operatorname{cone}(\{\tbinom{e}{0} \mid e \in E\}).$$

Aus Folgerung (1.20)(iii) ergibt sich dann:

$$\begin{aligned} P_1^\circ &= \{z \in \mathbb{K}^{n+1} \mid z^T u \le 0 \ \forall u \in P_1\} \\ &= \{z \in \mathbb{K}^{n+1} \mid z^T \tbinom{v}{1} \le 0 \ \forall v \in V, z^T \tbinom{e}{0} \le 0 \ \forall e \in E\} \\ &= \left\{z \mid \begin{pmatrix} V^T & \mathbb{1} \\ E^T & 0 \end{pmatrix} z \le 0\right\} = P\left(\begin{pmatrix} V^T & \mathbb{1} \\ E^T & 0 \end{pmatrix}, 0\right). \end{aligned}$$

Mit Folgerung (1.7) $P(A,0)^\circ = \operatorname{cone}(A^T)$ erhalten wir nun

$$\operatorname{hog}(P) = P_1^{\circ\circ} = P\left(\begin{pmatrix} V^T & \mathbb{1} \\ E^T & 0 \end{pmatrix}, 0\right)^\circ = \operatorname{cone}\begin{pmatrix} V & E \\ \mathbb{1}^T & 0^T \end{pmatrix}.$$

Die zweite Charakterisierung von $\operatorname{hog}(P)$ folgt aus einer anderen Darstellung von P_1°. Es gilt nämlich mit Satz (1.18):

$$\begin{aligned} P_1^\circ &= \{\tbinom{y}{\lambda} \in \mathbb{K}^{n+1} \mid y^T x + \lambda 1 \le 0 \ \forall x \in P\} = \{\tbinom{y}{\lambda} \in \mathbb{K}^{n+1} \mid y^T x \le -\lambda \ \forall x \in P\} \\ &= \{\tbinom{y}{\lambda} \in \mathbb{K}^{n+1} \mid \tbinom{y}{-\lambda} \in P^\gamma\} = \{\tbinom{y}{\lambda} \in \mathbb{K}^{n+1} \mid \tbinom{y}{-\lambda} \in \operatorname{cone}\begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix}\} \\ &= \operatorname{cone}\begin{pmatrix} A^T & 0 \\ -b^T & -1 \end{pmatrix}. \end{aligned}$$

Folgerung (1.10) impliziert nun

$$\operatorname{hog}(P) = P_1^{\circ\circ} = \left(\operatorname{cone}\begin{pmatrix} A^T & 0 \\ -b^T & -1 \end{pmatrix}\right)^\circ = P\left(\begin{pmatrix} A & -b \\ 0 & -1 \end{pmatrix}, 0\right). \qquad \square$$

In Abbildung 1.4 sind ein Polyeder $P \subseteq \mathbb{R}^1$, die im obigen Beweis definierte Menge P_1 und $\operatorname{hog}(P)$ dargestellt.

(1.46) Bemerkung. Sei $P \subseteq \mathbb{K}^n$ ein Polyeder, dann gilt:

(a) $x \in P \iff \binom{x}{1} \in \operatorname{hog}(P)$.

(b) $x \in \operatorname{rec}(P) \iff \binom{x}{0} \in \operatorname{hog}(P)$. $\triangle$

Beweis. (a) ist trivial.

(b) Sei $P = P(A,b)$ eine Darstellung von P, dann gilt $\operatorname{hog}(P) = P(B,0)$ mit $B = \begin{pmatrix} A & -b \\ 0 & -1 \end{pmatrix}$. Folglich gilt nach (1.45) und (1.43)

$$\begin{pmatrix} x \\ 0 \end{pmatrix} \in \operatorname{hog}(P) \iff \begin{pmatrix} x \\ 0 \end{pmatrix} \in P(B,0) \iff Ax \le 0 \iff x \in \operatorname{rec}(P). \qquad \square$$

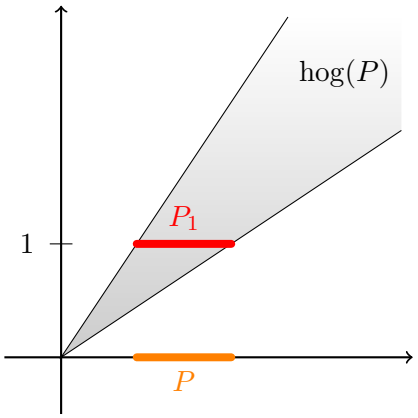


Abbildung 1.4: 1-dimensionaler Polyeder und seine Homogenisierung

1.7 Extremalen von spitzen Polyedern

Wir wollen nachfolgend einige Aussagen über spitze Polyeder beweisen, die sich – entsprechend modifiziert – auch für allgemeine Polyeder zeigen lassen. Dabei treten jedoch einige unschöne technische Komplikationen auf, so dass wir hier auf die Behandlung dieser Verallgemeinerung verzichten.

Wir erinnern daran, dass ein Polyeder spitz genannt wird, wenn es eine Ecke (nulldimensionale Seitenfläche) besitzt. Die folgende Aussage erweitert Satz (8.11) aus dem ADM I Skript.

(1.47) Satz. *Sei $P = P(A, b) \subseteq \mathbb{K}^n$ ein nichtleeres Polyeder, dann sind äquivalent:*

(1) *P ist spitz.*

(2) $\operatorname{rang}(A) = n$.

(3) $\operatorname{rec}(P)$ *ist spitz, d. h.* 0 *ist eine Ecke von* $\operatorname{rec}(P)$.

(4) *Jede nichtleere Seitenfläche von P ist spitz.*

(5) $\operatorname{hog}(P)$ *ist spitz.*

(6) *P enthält keine Gerade.*

(7) $\operatorname{rec}(P)$ *enthält keine Gerade.*

(8) $\operatorname{lineal}(P) = \{0\}$. △

Beweis. Die Äquivalenz von (1), (2) und (4) wurde schon in ADM I in Satz (8.11) gezeigt.

Aus der Äquivalenz von (1) und (2) folgt direkt die Äquivalenz der Aussagen (2), (3) und (5), da

$$\begin{aligned} \operatorname{rec}(P) &= P(A, 0), \text{ nach (1.43)}, \\ \operatorname{hog}(P) &= P\left(\begin{pmatrix} A & -b \\ 0 & -1 \end{pmatrix}, 0\right), \text{ nach (1.45)}. \end{aligned}$$

(3) $\Longrightarrow$ (6). Angenommen P enthält eine Gerade $G = \{u + \lambda v \mid \lambda \in \mathbb{K}\}$, $v \neq 0$, dann gilt $b \geq A(u + \lambda v) = Au + \lambda Av$ für alle $\lambda \in \mathbb{K}$. Daraus folgt $A(\lambda v) \leq 0$ für alle $\lambda \in \mathbb{K}$ und somit $v, -v \in \operatorname{rec}(P)$, d. h. $0 = \frac{1}{2}v + \frac{1}{2}(-v)$ ist eine Konvexkombination. Also ist 0 keine Ecke von $\operatorname{rec}(P)$.

(6) $\Longrightarrow$ (3). Ist $\operatorname{rec}(P)$ nicht spitz, so ist 0 echte Konvexkombination von Vektoren aus $\operatorname{rec}(P)$, sagen wir $0 = \lambda u + (1 - \lambda)v$, $u \neq 0 \neq v$, $0 < \lambda < 1$. Dann aber ist neben u auch $-\lambda u = (1 - \lambda)v \in \operatorname{rec}(P)$ und folglich ist $G = \{\mu u \mid \mu \in \mathbb{K}\}$ eine Gerade in $\operatorname{rec}(P)$, und für alle $x \in P$ ist $x + G$ eine Gerade in P.

Die Äquivalenz von (7) und (8) zu den übrigen Aussagen ist nun offensichtlich. □

Der folgende Hilfssatz über innere Punkte wird im Weiteren benötigt.

(1.48) Lemma. *Ist F eine nichtleere Seitenfläche von $P = P(A, b)$, gilt $I = \operatorname{eq}(F)$, und ist $B = \{y^1, \ldots, y^k\}$ eine Basis des Kerns von $A_{I.}$, dann gibt es zu jedem inneren Punkt $x \in F$ von F ein $\varepsilon > 0$, so dass $x \pm \varepsilon y^j \in P$ für alle $j = 1, \ldots, k$ gilt.* △

Beweis. Übungsaufgabe. □

(1.49) Definition. *Sei P ein Polyeder. Ein Vektor $z \in \operatorname{rec}(P) \setminus \{0\}$ heißt* Extremale *(oder* Extremalvektor*) von P, wenn $\operatorname{cone}(\{z\})$ ein Extremalstrahl von $\operatorname{rec}(P)$ ist.* △

Nur spitze Polyeder haben Extremalen. Denn ist P nicht spitz, so ist nach (1.47) $\operatorname{rec}(P)$ nicht spitz, also ist der Nullvektor eine echte Konvexkombination zweier von Null verschiedener Vektoren, sagen wir $0 = \lambda u + (1 - \lambda)v$, $0 < \lambda < 1$. Ist $F = \operatorname{cone}(\{z\})$ ein Extremalstrahl von $\operatorname{rec}(P)$, so gibt es eine bezüglich $\operatorname{rec}(P)$ gültige Ungleichung $c^T x \leq 0$ mit $F = \{x \in \operatorname{rec}(P) \mid c^T x = 0\}$. Nun gilt $0 = c^T 0 = c^T(\lambda u + (1 - \lambda)v) = \lambda c^T u + (1 - \lambda)c^T v \leq 0$. Aus $c^T u \leq 0$, $c^T v \leq 0$ folgt $c^T u = c^T v = 0$ und somit $u, v \in F$, ein Widerspruch. Aussagen über Extremalen machen also nur für spitze Polyeder Sinn.

Ist K speziell ein spitzer polyedrischer Kegel, so ist (wegen $\operatorname{rec}(K) = K$) eine Extremale von K ein Vektor $z \in K$, so dass $\operatorname{cone}(\{z\})$ ein Extremalstrahl von K ist. Das heißt, jeder auf einem Extremalstrahl von K gelegener und von Null verschiedener Vektor ist eine Extremale von K.

(1.50) Satz. *Seien $P = P(A, b) \subseteq \mathbb{K}^n$ ein spitzes Polyeder und $z \in \operatorname{rec}(P) \setminus \{0\}$. Dann sind äquivalent:*

(1) *z ist eine Extremale von P.*

(2) *$\operatorname{cone}(\{z\})$ ist ein Extremalstrahl von $\operatorname{rec}(P)$.*

(3) *z lässt sich nicht als echte konische Kombination zweier linear unabhängiger Elemente von* $\operatorname{rec}(P)$ *darstellen.*

(4) $(\operatorname{rec}(P) \setminus \operatorname{cone}\{z\}) \cup \{0\}$ *ist ein Kegel.*

(5) $\operatorname{rang}(A_{\operatorname{eq}(\{z\}).}) = n - 1$ *(wobei sich* eq *auf das System* $Ax \leq 0$ *bezieht).* △

Beweis. (1) $\iff$ (2). Definition.

(2) $\implies$ (3). Ist $F := \operatorname{cone}(\{z\})$ ein Extremalstrahl von $\operatorname{rec}(P)$, so ist F eine eindimensionale Seitenfläche von $\operatorname{rec}(P)$, d. h. F kann keine zwei linear unabhängigen Vektoren enthalten. Insbesondere gibt es eine bezüglich $\operatorname{rec}(P)$ gültige Ungleichung $c^T x \leq 0$ mit $F = \{x \in \operatorname{rec}(P) \mid c^T x = 0\}$. Gibt es zwei linear unabhängige Vektoren $u, v \in \operatorname{rec}(P)$ und $\lambda, \mu > 0$ mit $z = \lambda u + \mu v$, so gilt $0 = c^T z = c^T(\lambda u + \mu v) = \lambda c^T u + \mu c^T v \leq 0$. Daraus folgt $c^T u = c^T v = 0$, d. h. $u, v \in F$, ein Widerspruch.

(3) $\iff$ (4). Trivial.

(3) $\implies$ (5). Sei $I = \operatorname{eq}(\{z\})$, dann ist z innerer Punkt von $F := \{x \in \operatorname{rec}(P) \mid A_I.x = 0\}$. Ist $\operatorname{rang}(A_I.) < n - 1$, dann enthält der Kern von $A_I.$ einen von z linear unabhängigen Vektor u. Nach Lemma (1.48) gibt es ein $\varepsilon > 0$, so dass $z \pm \varepsilon u \in \operatorname{rec}(P)$ gilt. Dann aber gilt $z = \frac{1}{2}(z+\varepsilon u)+\frac{1}{2}(z-\varepsilon u)$, d. h. z ist echte konische Kombination von zwei linear unabhängigen Elementen von $\operatorname{rec}(P)$, Widerspruch. Offenbar ist $\operatorname{rang}(A_I.) \neq n$.

(5) $\implies$ (2). Sei $I = \operatorname{eq}(\{z\})$, dann folgt für $F = \{x \in \operatorname{rec}(P) \mid A_I.x = 0\}$ aus der Voraussetzung, dass $\dim(F) = 1$ gilt. Da $\operatorname{rec}(P)$ spitz ist, enthält nach (1.47) $\operatorname{rec}(P)$ keine Gerade, also muss die eindimensionale Seitenfläche F der Strahl $\operatorname{cone}(\{z\})$ sein. □

Wir wollen nun noch eine Beziehung zwischen den Ecken und Extremalen eines spitzen Polyeders und den Extremalen seiner Homogenisierung aufzeigen.

(1.51) Satz. *Sei* $P \subseteq \mathbb{K}^n$ *ein spitzes Polyeder, dann gilt:*

(a) *x ist Ecke von P* $\iff$ $\binom{x}{1}$ *ist Extremale von* $\operatorname{hog}(P)$.

(b) *z ist Extremale von P* $\iff$ $\binom{z}{0}$ *ist Extremale von* $\operatorname{hog}(P)$. △

Beweis. Sei $P = P(A, b)$, dann gilt nach Satz (1.45) $\operatorname{hog}(P) = P(B, 0)$ mit

$$B = \begin{pmatrix} A & -b \\ 0 & -1 \end{pmatrix}.$$

(a) Sei $I = \operatorname{eq}(\{x\})$ bezüglich $P(A, b)$. x ist Ecke von P $\iff$ $\operatorname{rang}(A_I.) = n$ (nach Satz (8.8) aus dem ADM I Skript) $\iff$ $\operatorname{rang}(B_{\operatorname{eq}(\{\binom{x}{1}\}).}) = n$ (denn die neu hinzugekommene Ungleichung ist nicht mit Gleichheit erfüllt) $\overset{(1.50)}{\iff}$ $\binom{x}{1}$ ist Extremale von $\operatorname{hog}(P)$.

(b) z ist Extremale von $P \iff z$ ist Extremale von $\mathrm{rec}(P) \overset{(1.50)}{\iff} \mathrm{rang}(A_{\mathrm{eq}(\{z\}).}) = n-1 \iff \mathrm{rang}(B_{\mathrm{eq}(\{\binom{z}{0}\}).}) = n \overset{(1.50)}{\iff} \binom{z}{0}$ ist Extremale von $\mathrm{hog}(P)$. □

1.8 Weitere Darstellungssätze

Wir knüpfen hier an Abschnitt 1.3 an, wo wir bereits verschiedene Darstellungssätze bewiesen haben. Einige dieser Sätze können wir mit Hilfe der nun gewonnenen Erkenntnisse verschärfen.

Ist K ein polyedrischer Kegel und gilt $K = \mathrm{cone}(E)$, dann nennen wir E eine *Kegelbasis von* K, wenn es keine echte Teilmenge E' von E gibt mit $K = \mathrm{cone}(E')$ und wenn jede andere minimale Menge F mit $K = \mathrm{cone}(F)$ dieselbe Kardinalität wie E hat. Ist P ein Polytop, dann heißt eine Menge V mit $P = \mathrm{conv}(V)$ *konvexe Basis von* P, wenn V keine echte Teilmenge besitzt, deren konvexe Hülle P ist, und wenn jede andere minimale Menge W mit $P = \mathrm{conv}(W)$ dieselbe Kardinalität wie V hat.

Trivialerweise sind die Elemente einer Kegelbasis E *konisch unabhängig*, d. h. kein $e \in E$ ist konische Kombination der übrigen Elemente von E; und die Elemente einer konvexen Basis sind *konvex unabhängig*, d. h. kein Element von V ist Konvexkombination der übrigen Elemente von V. Es gilt aber keineswegs, dass jeder Vektor $x \in \mathrm{cone}(E)$ bzw. $x \in \mathrm{conv}(V)$ eine eindeutige konische bzw. konvexe Darstellung durch Vektoren aus E bzw. V hat. In dieser Hinsicht unterscheiden sich also Kegelbasen und konvexe Basen von Vektorraumbasen.

(1.52) Satz. *Sei $\{0\} \neq K \subseteq \mathbb{K}^n$ ein spitzer polyedrischer Kegel, dann sind äquivalent:*

(1) *E ist eine Kegelbasis von K.*

(2) *E ist eine Menge, die man dadurch erhält, dass man aus jedem Extremalstrahl von K genau einen von Null verschiedenen Vektor (also eine Extremale von K) auswählt.*

△

Beweis. Ist z Extremale von K, so ist $K' := (K \setminus \mathrm{cone}(\{z\})) \cup \{0\}$ nach (1.50) ein Kegel. Folglich gilt $\mathrm{cone}(E) \subseteq K'$ für alle Teilmengen E von K'. Also muss jede Kegelbasis von K mindestens ein (aus der Basiseigenschaft folgt sofort „genau ein“) Element von $\mathrm{cone}(\{z\}) \setminus \{0\}$ enthalten.

Zum Beweis, dass jede wie in (2) spezifizierte Menge eine Kegelbasis ist, benutzen wir Induktion über $d = \dim K$. Für Kegel der Dimension 1 ist die Behauptung trivial. Sei die Behauptung für Kegel der Dimension d richtig und K ein Kegel mit $\dim K = d+1$. Sei $y \in K \setminus \{0\}$ beliebig und $c \in \mathbb{K}^n \setminus \{0\}$ ein Vektor, so dass die Ecke 0 von K die eindeutig bestimmte Lösung von $\max\{c^T x \mid x \in K\}$ ist (c existiert nach Satz (8.8) aus dem ADM I Skript). Sei $z \in \{x \mid c^T x = 0\} \setminus \{0\}$. Dann ist für die Gerade

$$G = \{y + \lambda z \mid \lambda \in \mathbb{K}\}$$

die Menge $K \cap G$ ein endliches Streckenstück (andernfalls wäre $z \in \text{rec}(K) = K$, und wegen $c^T z = 0$ wäre 0 nicht der eindeutige Maximalpunkt). Folglich gibt es zwei Punkte z_1 und z_2, die auf echten Seitenflächen, sagen wir F_1 und F_2, von K liegen, so dass $K \cap G = \text{conv}(\{z_1, z_2\})$. Die Dimensionen der Seitenflächen F_1, F_2 sind höchstens d, F_1 und F_2 sind Kegel, und die Extremalstrahlen von F_1 und F_2 sind Extremalstrahlen von K. Nach Induktionsvoraussetzung werden z_1 und z_2 durch die in (2) festgelegten Mengen bezüglich F_1 und F_2 konisch erzeugt. Daraus folgt, dass y durch jede Menge des Typs (2) konisch erzeugt werden kann. Dies impliziert die Behauptung. □

(1.53) Korollar. *Jeder spitze polyedrische Kegel besitzt eine – bis auf positive Skalierung der einzelnen Elemente – eindeutige Kegelbasis.* △

(1.54) Korollar. *Jeder spitze polyedrische Kegel $K \neq \{0\}$ ist die Summe seiner Extremalstrahlen, d. h. sind* $\text{cone}(\{e_i\})$, $i = 1, \ldots, k$ *die Extremalstrahlen von K, so gilt*

$$K = \text{cone}(\{e_1, \ldots, e_k\}) = \sum_{i=1}^{k} \text{cone}(\{e_i\}).$$

△

Der folgende Satz verschärft (1.13) für spitze Polyeder.

(1.55) Satz. *Jedes spitze Polyeder P lässt sich darstellen als die Summe der konvexen Hülle seiner Ecken und der konischen Hülle seiner Extremalen, d. h. sind V die Eckenmenge von P und* $\text{cone}(\{e\})$, $e \in E$, *die Extremalstrahlen von* $\text{rec}(P)$ *(bzw. ist E eine Kegelbasis von* $\text{rec}(P)$*), so gilt*

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

△

Beweis. Sei $\text{hog}(P)$ die Homogenisierung von P. Da P spitz ist, ist $\text{hog}(P)$ nach (1.47) ein spitzer Kegel. Nach (1.54) ist $\text{hog}(P)$ die Summe seiner Extremalstrahlen $\text{cone}(\{e_i'\})$, $i = 1, \ldots, k$. O. B. d. A. können wir annehmen, dass $e_i' = \binom{v_i}{1}$, $i = 1, \ldots, p$, und $e_i' = \binom{e_i}{0}$, $i = p+1, \ldots, k$ gilt. Aus (1.51) folgt: $V = \{v_1, \ldots, v_p\}$ ist die Eckenmenge von P und $E = \{e_{p+1}, \ldots, e_k\}$ ist die Extremalenmenge von P. Nach (1.46) gilt $x \in P \iff \binom{x}{1} \in \text{hog}(P)$ und somit folgt $x \in P \iff x = \sum_{i=1}^{p} \lambda_i v_i + \sum_{i=p+1}^{k} \mu_i e_i$, $\lambda_i, \mu_i \geq 0$, $\sum_{i=1}^{p} \lambda_i = 1$, d. h. $x \in \text{conv}(V) + \text{cone}(E)$. □

(1.56) Korollar (Satz von Krein-Milman). *Sei P ein Polyeder und V die Menge seiner Ecken, dann gilt*

$$P \text{ ist ein Polytop} \iff P = \text{conv}(V).$$

△

(1.57) Korollar. *Polytope haben eine eindeutige konvexe Basis.* △

Für die lineare Optimierung ist die folgende Beobachtung wichtig.

(1.58) Satz. *Seien $P \subseteq \mathbb{K}^n$ ein spitzes Polyeder und $c \in \mathbb{K}^n$. Das lineare Programm $\max c^T x$, $x \in P$ ist genau dann unbeschränkt, wenn es eine Extremale e von P gibt mit $c^T e > 0$.* △

Beweis. Seien V die Eckenmenge von P und E eine Kegelbasis von $\mathrm{rec}(P)$, dann gilt nach (1.55) $P = \mathrm{conv}(V) + \mathrm{cone}(E)$. Es ist $\gamma := \max\{c^T v \mid v \in V\} < \infty$, da V endlich und $\gamma = \max\{c^T x \mid x \in \mathrm{conv}(V)\}$. Gilt $c^T e \leq 0 \ \forall e \in E$, so ist $\gamma = \max\{c^T x \mid x \in P\} < \infty$. Falls also $\max\{c^T x \mid x \in P\}$ unbeschränkt ist, muss für mindestens ein $e \in E$ gelten $c^T e > 0$. Die umgekehrte Richtung ist trivial. □

Wie bereits bemerkt, haben Elemente von spitzen Polyedern P i. A. keine eindeutige Darstellung als konvexe und konische Kombination von Ecken und Extremalen. Man kann jedoch zeigen, dass zu einer derartigen Darstellung von Elementen von P nicht allzu viele Ecken und Extremalen benötigt werden.

(1.59) Satz. *Es seien $K \subseteq \mathbb{K}^n$ ein spitzer Kegel und $0 \neq x \in K$. Dann gibt es Extremalen $y_1, \ldots, y_d$ von K, wobei $d \leq \dim(K) \leq n$ gilt, mit*

$$x = \sum_{i=1}^{d} y_i.$$

△

Beweis. Es seien $\mathrm{cone}(\{e_i\})$ die Extremalstrahlen von K, $i = 1, \ldots, k$, dann gibt es nach (1.54) Skalare $\lambda_i \geq 0$, $i = 1, \ldots, k$, mit

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

Unter allen möglichen Darstellungen von x dieser Art sei die obige eine, so dass $I = \{i \in \{1, \ldots, k\} \mid \lambda_i > 0\}$ minimal ist. Sagen wir, es gilt $I = \{1, \ldots, d\}$. Angenommen die Vektoren e_i, $i \in I$ sind linear abhängig, dann gibt es $\mu_1, \ldots, \mu_d \in \mathbb{K}$, nicht alle μ_i Null, so dass gilt

$$\sum_{i=1}^{d} \mu_i e_i = 0.$$

Angenommen $\mu_i \geq 0$ für $i = 1, \ldots, d$, und o. B. d. A. sei $\mu_1 > 0$. Dann ist $-e_1 = \sum_{i=2}^{d} \frac{\mu_i}{\mu_1} e_i$ eine konische Kombination, und nach Satz (1.44) gilt $e_1 \in \mathrm{lineal}(K)$. Dies widerspricht nach (1.47) der Voraussetzung K ist spitz.

O. B. d. A. können wir daher annehmen, dass gilt $\mu_1 < 0$ und

$$\frac{\lambda_1}{\mu_1} \ = \ \max\left\{\frac{\lambda_i}{\mu_i} \mid \mu_i < 0\right\}.$$

Daraus folgt

$$e_1 = -\sum_{i=2}^{d} \frac{\mu_i}{\mu_1} e_i \ , \qquad x = \sum_{i=2}^{d} \big(\lambda_i - \frac{\lambda_1}{\mu_1}\mu_i\big) e_i.$$

Diese Darstellung von x ist eine konische Kombination, denn

$$\mu_i \geq 0 \implies \lambda_i - \frac{\lambda_1}{\mu_1}\mu_i \geq 0,$$
$$\mu_i < 0 \implies \frac{\lambda_i}{\mu_i} \leq \frac{\lambda_1}{\mu_1} \implies \lambda_i \geq \frac{\lambda_1}{\mu_1}\mu_i,$$

also kann x mit weniger als d Extremalen konisch dargestellt werden, Widerspruch zur Minimalität! Da ein Kegel höchstens $\dim(K)$ linear unabhängige Vektoren enthält, folgt $d \leq \dim(K)$. Setzen wir $y_i = \lambda_i e_i$, $i = 1, \ldots, d$, so sind die Vektoren y_i Extremalen von K mit der gewünschten Eigenschaft. □

(1.60) Korollar. *Ist $P \subseteq \mathbb{K}^n$ ein spitzes Polyeder und ist $x \in P$, dann gibt es Ecken $v_0, v_1, \ldots, v_k$ und Extremalen $e_{k+1}, e_{k+2}, \ldots, e_d$ von P mit $d \leq \dim(P)$ und nichtnegative Skalare $\lambda_0, \ldots, \lambda_k$ mit $\sum_{i=0}^{k} \lambda_i = 1$, so dass gilt*

$$x = \sum_{i=0}^{k} \lambda_i v_i + \sum_{i=k+1}^{d} e_i.$$

△

Beweis. Nach (1.47) ist $\mathrm{hog}(P)$ ein spitzer Kegel, und die Dimension von $\mathrm{hog}(P)$ ist $\dim(P)+1$. Nach (1.46) gilt $x \in P \iff \binom{x}{1} \in \mathrm{hog}(P)$. Nach Satz (1.59) ist $\binom{x}{1}$ konische Kombination von $d+1 \leq \dim(P)+1$ Extremalen von $\mathrm{hog}(P)$. O. B. d. A. können wir annehmen, dass gilt

$$\binom{x}{1} = \sum_{i=0}^{k} \binom{y_i}{\lambda_i} + \sum_{i=k+1}^{d} \binom{e_i}{0},$$

wobei $\lambda_i > 0$, $i = 0, \ldots, k$. Für $v_i := \frac{1}{\lambda_i} y_i$ gilt dann

$$x = \sum_{i=0}^{k} \lambda_i v_i + \sum_{i=k+1}^{d} e_i, \qquad \sum_{i=0}^{k} \lambda_i = 1.$$

Ferner sind nach (1.51) die Vektoren v_i Ecken von P und die Vektoren e_i Extremalen von P. Also haben wir die gewünschte Kombination von x gefunden. □

Das folgende direkte Korollar von (1.60) wird in der Literatur häufig **Satz von Caratheodory** genannt.

(1.61) Korollar. *Ist $P \subseteq \mathbb{K}^n$ ein Polytop, dann ist jedes Element von P Konvexkombination von höchstens $\dim(P)+1$ Ecken von P.* △

Literaturverzeichnis

G. Averkov und L. Bröcker. Minimal polynomial descriptions of polyhedra and special semialgebraic sets. *Advances in Geometry*, 12(3):447–459, 2012.

A. Bachem und M. Grötschel. New aspects of polyhedral theory. In B. Korte, editor, *Modern Applied Mathematics – Optimization and Operations Research*. North-Holland, Amsterdam, 1982.

J. Bochnak, M. Coste, und M.-F. Roy. *Real algebraic geometry*. Springer, 1998.

H. Bosse, M. Grötschel, und M. Henk. Polynomial inequalities representing polyhedra. *Mathematical Programming*, 103(1):35–44, 2005.

B. Grünbaum. *Convex Polytopes*, Band 221 von *Graduate Texts in Mathematics*. Springer, 2nd edition, 2003.

J. Matoušek. *Lectures on Discrete Geometry*, Band 212 von *Graduate Texts in Mathematics*. Springer, 2002.

G. M. Ziegler. *Lectures on Polytopes*, Band 152 von *Graduate Texts in Mathematics*. Springer, 2010.

2 Matroide und Unabhängigkeitssysteme

Wir werden nun einige sehr allgemeine Klassen kombinatorischer Optimierungsprobleme kennenlernen. Diese enthalten die in den Kapiteln 5 bis 7 in ADM I betrachteten Probleme. Der in Kapitel 5 von ADM I eingeführte Greedy-Algorithmus wird hier von besonderer Bedeutung sein. Allerdings treten bei der algorithmischen Behandlung der in diesem Kapitel besprochenen allgemeinen Probleme einige Schwierigkeiten auf, die wir eingehend darstellen und diskutieren werden.

Gute Bücher über Matroidtheorie sind Oxley (1992), Truemper (1992) und Welsh (1976), ein lesenswerter Übersichtsartikel ist Welsh (1995). Optimierungsprobleme auf Matroiden werden z. B. in Bixby und Cunningham (1995) und Lee (2004), Seiten 49–74, ausführlich behandelt. Eine umfassende Übersicht über Matroide, submodulare Funktionen und verwandte Strukturen findet sich in Part IV, Volume B von Schrijver (2003) auf den Seiten 649–852.

2.1 Allgemeine Unabhängigkeitssysteme

E sei im folgenden immer eine endliche Menge, 2^E bezeichne die Menge aller Teilmengen von E.

(2.1) Definition. *Eine Menge $\mathcal{I} \subseteq 2^E$ heißt* Unabhängigkeitssystem *(oder* monotones Mengensystem*) auf E, wenn $\mathcal{I}$ die folgenden Axiome erfüllt:*

(I.1) $\emptyset \in \mathcal{I}$,

(I.2) $F \subseteq G \in \mathcal{I} \Longrightarrow F \in \mathcal{I}$.

Häufig wird auch das Paar $(E, \mathcal{I})$ Unabhängigkeitssystem genannt. Die Teilmengen von E, die in $\mathcal{I}$ enthalten sind, heißen unabhängige Mengen*, alle übrigen Teilmengen von E heißen* abhängige Mengen. △

Mit jedem Unabhängigkeitssystem $(E, \mathcal{I})$ sind auf kanonische Weise andere Mengensysteme verbunden, die wir nun kurz einführen wollen.

Die bezüglich Mengeninklusion minimalen abhängigen Teilmengen von E heißen *Zirkuits* (oder *Kreise*), d. h. $C \subseteq E$ ist ein Zirkuit, wenn C abhängig ist und wenn C keine von sich selbst verschiedene abhängige Teilmenge enthält. Die Menge aller Zirkuits (bzgl. eines Unabhängigkeitssystems $\mathcal{I}$) heißt *Zirkuitsystem* und wird mit $\mathcal{C}$ bezeichnet.

Ist $F \subseteq E$, so heißt jede Teilmenge von F, die unabhängig ist und die in keiner anderen unabhängigen Teilmenge von F enthalten ist, *Basis* von F, d. h.

$$B \text{ Basis von } F \iff (B, B' \in \mathcal{I}, B \subseteq B' \subseteq F \Longrightarrow B = B').$$

Die Menge aller Basen der Grundmenge E heißt *Basissystem* (bzgl. $\mathcal{I}$) und wird mit $\mathcal{B}$ bezeichnet.

Für jede Menge $F \subseteq E$ heißt die ganze Zahl

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

Rang von F. Die Rangfunktion r ist also eine Funktion, die 2^E in die nicht-negativen ganzen Zahlen abbildet.

Offenbar induziert jedes Unabhängigkeitssystem $\mathcal{I}$ auf E ein eindeutig bestimmtes Zirkuitsystem, ein eindeutig bestimmtes Basissystem und eine eindeutig bestimmte Rangfunktion. Es gilt auch die Umkehrung, wie wir nachfolgend (ohne Beweis) skizzieren.

Zirkuitsysteme und Basissysteme sind nach Definition *Antiketten (Clutter)*, d. h. Systeme von Mengen, so dass keine zwei Mengen ineinander enthalten sind.

Ist $\mathcal{B} \neq \emptyset$ eine Antikette auf E, so ist

$$\mathcal{I} := \{I \subseteq E \mid \exists B \in \mathcal{B} \text{ mit } I \subseteq B\}$$

ein Unabhängigkeitssystem auf E, und $\mathcal{B}$ ist das zu $\mathcal{I}$ gehörige Basissystem.

Ist $\mathcal{C} \neq \{\emptyset\}$ eine Antikette auf E, so ist

$$\mathcal{I} := \{I \subseteq E \mid I \text{ enthält kein Element von } \mathcal{C}\} \tag{2.2}$$

ein Unabhängigkeitssystem, und $\mathcal{C}$ ist das zu $\mathcal{I}$ gehörige Zirkuitsystem.

Die oben definierte Rangfunktion hat folgende Eigenschaften. Sie ist *subkardinal*, d. h. für alle $F \subseteq E$ gilt

$$r(F) \leq |F|,$$

sie ist *monoton*, d. h. für alle $F, G \subseteq E$ gilt

$$F \subseteq G \Longrightarrow r(F) \leq r(G),$$

und sie ist *stark subadditiv*, d. h. für alle $F \subseteq E$, für alle ganzen Zahlen $k \geq 1$, für alle endlichen Indexmengen $K \subseteq \mathbb{N}$ und für alle Familien $(F_i)_{i \in K}$ von Teilmengen von F mit der Eigenschaft, dass $|\{i \in K \mid e \in F_i\}| = k$ für alle $e \in F$, gilt

$$k \cdot r(F) \leq \sum_{i \in K} r(F_i).$$

Ist $r : 2^E \to \mathbb{Z}_+$ eine subkardinale, monotone, stark subadditive Funktion, so ist

$$\mathcal{I} := \{I \subseteq E \mid r(I) = |I|\}$$

ein Unabhängigkeitssystem, dessen Rangfunktion die Funktion r ist.

Unabhängigkeitssysteme $\mathcal{I}$ auf einer Grundmenge E definieren also mathematische Strukturen, die äquivalent durch Zirkuitsysteme, Basissysteme oder Rangfunktionen gegeben werden können. Unabhängigkeitssysteme sind sehr allgemeine Objekte und besitzen zu wenig Struktur, um wirklich tiefliegende Aussagen über sie machen zu können.

Sei für jedes Element $e \in E$ ein Gewicht $c_e \in \mathbb{K}$ gegeben. Für $F \subseteq E$ setzen wir wie üblich

$$c(F) := \sum_{e \in F} c_e.$$

Das Problem, eine unabhängige Menge $I^* \in \mathcal{I}$ zu finden, so dass $c(I^*)$ maximal ist, heißt *Optimierungsproblem über einem Unabhängigkeitssystem* $\mathcal{I}$, d. h. wir suchen

$$\max\{c(I) \mid I \in \mathcal{I}\}. \tag{2.3}$$

Offenbar macht es hier keinen Sinn, Gewichte c_e zu betrachten, die nicht positiv sind. Denn wenn $I^* \in \mathcal{I}$ optimal ist, gilt $I' := I^* \setminus \{e \in E \mid c_e \leq 0\} \in \mathcal{I}$ (wegen (I.2)) und $c(I') \geq c(I^*)$, also ist I' ebenfalls eine optimale unabhängige Menge. Deswegen werden wir uns im Weiteren bei Optimierungsproblemen über Unabhängigkeitssystemen auf Gewichtsfunktionen beschränken, die positiv oder nicht-negativ sind.

Bei Optimierungsproblemen über Basissystemen ist es dagegen auch sinnvoll, negative Gewichte zuzulassen. Ist ein Basissystem $\mathcal{B}$ auf der Grundmenge E gegeben und sind $c_e \in \mathbb{K}$ Gewichte, so nennen wir

$$\min\{c(B) \mid B \in \mathcal{B}\} \tag{2.4}$$

Optimierungsproblem über einem Basissystem $\mathcal{B}$.

(2.5) Beispiel.

(a) Sei $G = (V, E)$ ein Graph. Eine Knotenmenge $S \subseteq V$ heißt *stabil (Clique)*, falls je zwei Knoten aus S nicht benachbart (benachbart) sind. Die Menge der stabilen Knotenmengen (Cliquen) ist ein Unabhängigkeitssystem auf V. Die Aufgabe – bei gegebenen Knotengewichten – eine gewichtsmaximale stabile Menge (Clique) zu finden, ist ein Optimierungsproblem über einem Unabhängigkeitssystem, vergleiche ADM I (3.13).

(b) Ein *Wald* in einem Graphen $G = (V, E)$ ist eine Kantenmenge, die keinen Kreis enthält. Die Menge aller Wälder bildet ein Unabhängigkeitssystem auf E. Das Problem, einen maximalen Wald in G zu finden, war Hauptthema von ADM I, Abschnitt 5.2. Offenbar ist in einem zusammenhängenden Graphen G die Menge der aufspannenden Bäume genau die Menge der Basen des Unabhängigkeitssystems der Wälder. Das Problem, einen minimalen aufspannenden Baum zu finden (siehe ADM I, Satz (5.13)), ist somit ein Optimierungsproblem über einem Basissystem.

(c) Ein *Matching* in einem Graphen $G = (V, E)$ ist eine Kantenmenge $M \subseteq E$, so dass jeder Knoten aus V in höchstens einer Kante aus M enthalten ist. Die Menge aller Matchings ist ein Unabhängigkeitssystem auf E. Das Matchingproblem ADM I, (3.10) ist also ein Optimierungsproblem über einem Unabhängigkeitssystem. Die Aufgabe, in einem vollständigen Graphen mit Kantengewichten ein minimales perfektes Matching zu finden, ist ein Optimierungsproblem über einem Basissystem. Analog bilden die zulässigen Lösungen eines 1-kapazitierten b-Matchingproblems ein Unabhängigkeitssystem.

(d) Gegeben sei ein vollständiger Digraph $D_n = (V, A)$ mit n Knoten und Bogenlängen c_{ij} für alle $(i, j) \in A$. Eine *Tour (gerichteter Hamiltonkreis)* ist ein gerichteter Kreis in D_n, der jeden Knoten enthält. Die Aufgabe, eine Tour mit minimalem Gewicht zu finden, heißt *asymmetrisches Travelling-Salesman-Problem*, siehe ADM I, (3.12). Die Menge $\mathcal{T}$ aller Touren ist kein Unabhängigkeitssystem, jedoch eine Antikette, also Basissystem eines Unabhängigkeitssystems. Das asymmetrische TSP ist somit ein Optimierungsproblem über einem Basissystem. Wir können es aber auch als Optimierungsproblem über einem Unabhängigkeitssystem auffassen. Dies geht wie folgt: Setzen wir

$$\tilde{\mathcal{T}} := \{I \subseteq A \mid \exists T \in \mathcal{T} \text{ mit } I \subseteq T\},$$
$$c'_{ij} := \max\{|c_{ij}| \mid (i, j) \in A\} + 1 - c_{ij},$$

so ist $\tilde{\mathcal{T}}$ ein Unabhängigkeitssystem, und jede Lösung von $\max\{c'(I) \mid I \in \tilde{\mathcal{T}}\}$ ist eine Tour, die – bezüglich der Gewichte c_{ij} – minimales Gewicht hat. (Auf die gleiche Weise kann man viele andere Optimierungsprobleme über Basissystemen in Optimierungsprobleme über Unabhängigkeitssystemen überführen.) Ebenso ist das symmetrische TSP ein Optimierungsproblem über einem Basissystem.

(e) Gegeben sei ein gerichteter Graph $D = (V, A)$. Ein *Branching* in D ist eine Bogenmenge $B \subseteq A$, die keinen Kreis (im ungerichteten Sinne) enthält, und die die Eigenschaft hat, dass jeder Knoten $v \in V$ Endknoten von höchstens einem Bogen aus B ist. Die Menge aller Branchings ist ein Unabhängigkeitssystem auf A. Das Problem, in einem vollständigen Digraphen eine minimale aufspannende Arboreszenz zu finden, ist ein Optimierungsproblem über einem Basissystem, siehe ADM I, (3.11). △

Überlegen Sie sich, welche der übrigen Beispiele aus ADM I, Abschnitt 3.3 als Optimierungsprobleme über Unabhängigkeits- oder Basissystemen aufgefasst werden können und welche nicht.

Verschiedene praktische Fragestellungen führen auch zu Optimierungsproblemen über Zirkuits. Sind die Elemente $e \in E$ durch Gewichte c_e bewertet, so kann man das Problem untersuchen, ein Zirkuit $C \in \mathcal{C}$ zu finden, das minimales Gewicht $c(C)$ hat. Die Aufgabe, in einem Graphen einen kürzesten Kreis zu bestimmen, ist z. B. von diesem Typ.

Allgemeiner noch ist folgende Frage von Interesse. Wir sagen, dass eine Menge $Z \subseteq E$ ein *Zyklus* ist, wenn Z die Vereinigung von paarweise disjunkten Zirkuits ist, d. h. wenn es Zirkuits $C_1, \ldots, C_k$ gibt mit $C_i \cap C_j = \emptyset$, $1 \leq i < j \leq k$, so dass $Z = \bigcup_{i=1}^{k} C_i$. Sind die Elemente $e \in E$ mit Gewichten c_e belegt, so sucht man nach einem Zyklus maximalen Gewichts. Das Chinesische Postbotenproblem (siehe ADM I, (3.12)) und das Max-Cut-Problem (siehe ADM I, (3.15)) sind z. B. von diesem Typ. Aus Zeitgründen können wir auf Optimierungsprobleme über Zirkuits bzw. Zyklen nicht eingehen, siehe hierzu Barahona und Grötschel (1986) und Grötschel und Truemper (1989).

2.2 Matroide

Wie die Beispiele aus dem vorigen Abschnitt zeigen, enthalten Optimierungsprobleme über Unabhängigkeitssystemen sowohl polynomial lösbare als auch $\mathcal{NP}$-vollständige Probleme. Man wird daher nicht erwarten können, dass für diese Probleme eine „gute“ Lösungstheorie existiert. Wir wollen nun eine Spezialklasse von Unabhängigkeitssystemen einführen, für die es so etwas gibt. In einem noch zu präzisierenden Sinn (siehe Folgerung (2.15)) ist dies die Klasse der Unabhängigkeitssysteme, für die der Greedy-Algorithmus (siehe ADM I, (5.7)) eine Optimallösung liefert.

(2.6) Definition. *Ein* Matroid M *besteht aus einer Grundmenge* E *zusammen mit einem Unabhängigkeitssystem* $\mathcal{I} \subseteq 2^E$*, das eine der folgenden äquivalenten Bedingungen erfüllt:*

$$\text{(I.3)} \quad I, J \in \mathcal{I}, |I| = |J| - 1 \implies \exists j \in J \setminus I \text{ mit } I \cup \{j\} \in \mathcal{I},$$

$$\text{(I.3}') \quad I, J \in \mathcal{I}, |I| < |J| \implies \exists K \subseteq J \setminus I \text{ mit } |I \cup K| = |J|, \text{ so dass } I \cup K \in \mathcal{I},$$

$$\text{(I.3}'') \quad F \subseteq E \text{ und } B, B' \text{ Basen von } F \implies |B| = |B'|. \qquad \triangle$$

Das heißt also, das Unabhängigkeitssystem eines Matroids auf E ist ein Mengensystem $\mathcal{I} \subseteq 2^E$, das die Axiome (I.1), (I.2) und eines der Axiome (I.3), (I.3′), (I.3″) erfüllt. Ein solches System erfüllt automatisch auch die beiden übrigen der drei Axiome (I.3), (I.3′), (I.3″).

Ein Wort zur Terminologie! Nach der obigen Definition ist ein Matroid M ein Paar $(E, \mathcal{I})$ mit den oben aufgeführten Eigenschaften. Wenn klar ist, um welche Grundmenge E es sich handelt, spricht man häufig auch einfach von dem Matroid $\mathcal{I}$, ohne dabei E explizit zu erwähnen. Man sagt auch, M (bzw. $\mathcal{I}$) ist ein Matroid auf E.

In Definition (2.6) haben wir von den „äquivalenten Bedingungen“ (I.3), (I.3′), (I.3″) gesprochen. Diese Äquivalenz muss natürlich bewiesen werden. Da wir hier jedoch keine Vorlesung über Matroide halten wollen, können wir auf die Beweise (aus Zeitgründen) nicht eingehen. Das gleiche gilt für die nachfolgend gemachten Aussagen über Zirkuit- und Basissysteme und Rangfunktionen. Beweise der hier gemachten Aussagen findet der interessierte Leser z. B. in Oxley (1992) und Welsh (1976).

Wie wir bereits gesehen haben, können Unabhängigkeitssysteme über Zirkuits, Basen oder Rangfunktionen beschrieben werden. Ist $M = (E, \mathcal{I})$ ein Matroid und sind $\mathcal{C}, \mathcal{B}, r$ das zugehörige Zirkuit-, Basissystem bzw. die zugehörige Rangfunktion, so ist natürlich $\mathcal{I}$ durch die Angabe von $\mathcal{C}$, $\mathcal{B}$ oder r eindeutig beschrieben. Gelegentlich werden daher auch die Paare $(E, \mathcal{C})$, $(E, \mathcal{B})$, (E, r) als Matroide bezeichnet, meistens dann, wenn es bei einer speziellen Untersuchung sinnvoller erscheint, mit Zirkuits, Basen oder Rangfunktionen statt mit Unabhängigkeitssystemen zu arbeiten.

Sicherlich induzieren nicht alle Zirkuit- oder Basissysteme oder alle Rangfunktionen Unabhängigkeitssysteme von Matroiden. Diese Antiketten bzw. Funktionen müssen spezielle Eigenschaften haben, damit das zugehörige Unabhängigkeitssystem das Axiom (I.3) erfüllt. Einige solcher Eigenschaften wollen wir kurz auflisten.

(2.7) Satz.

(a) *Eine Antikette $\mathcal{C} \subseteq 2^E$, $\mathcal{C} \neq \{\emptyset\}$, ist das* **Zirkuitsystem eines Matroids** *auf E genau dann, wenn eine der beiden folgenden äquivalenten Bedingungen erfüllt ist:*

(C.1) $C_1, C_2 \in \mathcal{C}$, $C_1 \neq C_2$, $z \in C_1 \cap C_2 \implies \exists C_3 \in \mathcal{C}$ *mit* $C_3 \subseteq (C_1 \cup C_2) \setminus \{z\}$,

(C.1′) $C_1, C_2 \in \mathcal{C}$, $C_1 \neq C_2$, $y \in C_1 \setminus C_2 \implies \forall x \in C_1 \cap C_2\ \exists C_3 \in \mathcal{C}$ *mit* $y \in C_3 \subseteq (C_1 \cup C_2) \setminus \{x\}$.

(b) *Eine Antikette $\mathcal{B} \subseteq 2^E$, $\mathcal{B} \neq \emptyset$, ist das* **Basissystem eines Matroids** *auf E genau dann, wenn das folgende Axiom erfüllt ist:*

(B.1) $B_1, B_2 \in \mathcal{B}$, $x \in B_1 \setminus B_2 \implies \exists y \in B_2 \setminus B_1$ *mit* $(B_1 \cup \{y\}) \setminus \{x\} \in \mathcal{B}$.

(c) *Eine Funktion $r : 2^E \to \mathbb{Z}$ ist die* **Rangfunktion eines Matroids** *auf E genau dann, wenn eines der beiden folgenden äquivalenten Axiomensysteme erfüllt ist:*

(R.1) $r(\emptyset) = 0$,

(R.2) $F \subseteq E$, $e \in E \implies r(F) \leq r(F \cup \{e\}) \leq r(F) + 1$,

(R.3) $F \subseteq E$, $f, g \in E$ *mit* $r(F \cup \{g\}) = r(F \cup \{f\}) = r(F) \implies r(F \cup \{g, f\}) = r(F)$,

beziehungsweise

(R.1′) $F \subseteq E \implies 0 \leq r(F) \leq |F|$ *(r ist subkardinal),*

(R.2′) $F \subseteq G \subseteq E \implies r(F) \leq r(G)$ *(r ist monoton),*

(R.3′) $F, G \subseteq E \implies r(F \cup G) + r(F \cap G) \leq r(F) + r(G)$ *(r ist submodular).* △

Es gibt noch einige hundert weitere äquivalente Definitionen von Matroiden (die Äquivalenz ist – auch in den obigen Fällen – nicht immer offensichtlich). Wir wollen uns jedoch mit den obigen begnügen.

Ist $\mathcal{I}_1$ ein Matroid auf einer Grundmenge E_1 und $\mathcal{I}_2$ ein Matroid auf E_2, so heißen $\mathcal{I}_1$ und $\mathcal{I}_2$ *isomorph*, falls es eine bijektive Abbildung $\varphi : E_1 \to E_2$ gibt mit

$$\varphi(F) \text{ ist unabhängig in } \mathcal{I}_2 \iff F \text{ ist unabhängig in } \mathcal{I}_1.$$

Eine Teilmenge $F \subseteq E$ heißt *abgeschlossen (bezüglich $\mathcal{I}$)*, falls gilt

$$r(F) < r(F \cup \{e\}) \text{ für alle } e \in E \setminus F.$$

Die Matroidtheorie kann man als eine gemeinsame Verallgemeinerung gewisser Aspekte der Graphentheorie und der linearen Algebra ansehen. Die beiden Beispiele, aus denen die Matroidtheorie entstanden ist, wollen wir daher zuerst vorstellen.

(2.8) Beispiel.

(a) **Graphische Matroide**
Das in (2.5)(b) definierte Unabhängigkeitssystem der Wälder eines Graphen $G = (V, E)$ ist ein Matroid. Ist $F \subseteq E$, so zerfällt (V, F) in Zusammenhangskomponenten $G_1 = (V_1, F_1), \ldots, G_k = (V_k, F_k)$. Die Basen der Zusammenhangskomponenten $G_1, \ldots, G_k$ sind die aufspannenden Bäume von $G_1, \ldots, G_k$. Jede Vereinigung von aufspannenden Bäumen von $G_1, \ldots, G_k$ ist eine Basis von F. Die Zirkuits von $\mathcal{I}$ sind die Kreise des Graphen G (daher der Name!). Der Rang einer Menge $F \subseteq E$ ist gegeben durch

$$r(F) = |V(F)| - \text{ Anzahl } k \text{ der Komponenten von } (V(F), F),$$

wobei $V(F)$ die Menge aller Knoten $v \in V$ bezeichnet, die in mindestens einer Kante aus F enthalten sind, vergleiche ADM I, Korollar (5.5).
Die Matroide, die wie oben angegeben auf einem Graphen definiert werden können (bzw. isomorph zu solchen sind), heißen *graphische Matroide*.

(b) **Matrix-Matroide**
Sei $A = (a_{ij})$ eine (m, n)-Matrix über einem beliebigen Körper K mit Spaltenvektoren $A_{\cdot 1}, \ldots, A_{\cdot n}$. $E = \{1, \ldots, n\}$ sei die Menge der Spaltenindizes von A. Eine Teilmenge $F \subseteq E$ heißt unabhängig, wenn die Vektoren $A_{\cdot j}$, $j \in F$, linear unabhängig in K^m sind. Da jede Teilmenge einer linear unabhängigen Menge wiederum linear unabhängig ist, ist das so definierte Mengensystem $\mathcal{I}$ offenbar ein Unabhängigkeitssystem. Die Menge aller Basen von E ist die Menge aller $B \subseteq E$, so dass die Vektoren $A_{\cdot j}$, $j \in B$, eine Basis des durch die Spaltenvektoren von A aufgespannten linearen Teilraums von K^m bilden. Das Basisaxiom (B.1) ist in diesem Falle aufgrund des Steinitz'schen Austauschsatzes erfüllt. (Dieser Satz war die Motivation für (B.1).)
Matroide, die auf die hier angegebene Weise konstruiert werden können, heißen *Matrix-Matroide* oder *lineare Matroide*. Die Rangfunktion von $\mathcal{I}$ entspricht der aus der linearen Algebra bekannten Rangfunktion des K^m beschränkt auf die Spalten von A. Ist $\mathcal{I}$ ein Matroid auf E und gibt es einen Körper K und eine (m, n)-Matrix A über K, so dass $\mathcal{I}$ zu dem Matrix-Matroid bezüglich A isomorph ist, dann heißt $\mathcal{I}$ *repräsentierbar (über K)*. (Natürlich kann man hier etwas verallgemeinern und anstelle von Körpern Schiefkörper oder andere geeignete Objekte betrachten und Repräsentierbarkeit über diesen studieren.) Matroide, die über dem zweielementigen Körper $GF(2)$ repräsentierbar sind, heißen *binär*. Eine umfassende Untersuchung dieser wichtigen Klasse von Matroiden findet sich in Truemper (1992). Matroide, die über allen Körpern repräsentierbar sind, nennt man *regulär*.

(c) **Binäre Matroide**
Die über dem zweielementigen Körper $GF(2)$ repräsentierbaren Matroide kann man auf sehr einfache Weise durch eine 0/1-Matrix repräsentieren. Seien M ein binäres Matroid auf E, T eine Basis von E und $S := E \setminus T$. Wir können o. B. d. A. annehmen, dass $T = \{e_1, \ldots, e_m\}$ und $S = \{e_{m+1}, \ldots, e_n\}$ gilt. Eine das Matroid M (mit Rang

m) repräsentierende Matrix A erhält man wie folgt: A hat m Zeilen und n Spalten. Die i-te Zeile „repräsentiert“ das i-te Basiselement e_i, $1 \leq i \leq m$, die j-te Spalte das j-te Element e_j, $1 \leq j \leq n$, von E; A hat die folgende Form (genannt *Standardform* oder *Standardrepräsentation*):

$$A = (I_m, B),$$

wobei I_m die (m, m)-Einheitsmatrix ist. Die j-te Spalte von A, $m + 1 \leq j \leq n$, wird wie folgt konstruiert. Fügt man das Element e_j zur Basis T hinzu, so kann man beweisen, dass genau ein Zirkuit C entsteht, genannt das Fundamentalzirkuit zu e_j. Für das Element a_{ij} von A definieren wir dann: $a_{ij} = 1$, falls das Basiselement e_i zu C gehört, $a_{ij} = 0$ sonst. Für das graphische Matroid des Beispielgraphen in Abbildung 2.1 ergibt sich so die folgende Standardrepräsentation:

	1	2	3	4	5	6	7	8	9	10
1	1					0	0	0	0	0
2		1			0	0	1	1	1	0
3			1			0	1	0	1	0
4	0			1		1	1	1	0	0
5					1	1	1	0	0	0

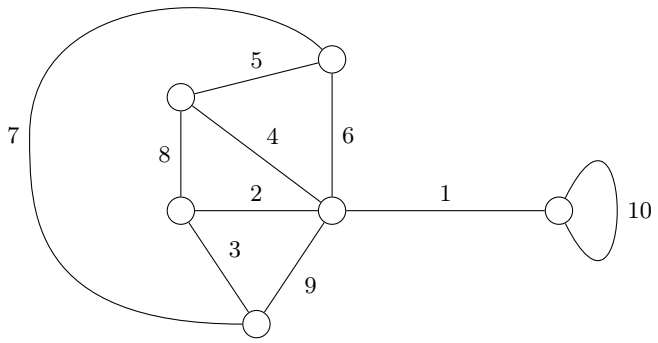


Abbildung 2.1: Graph mit 10 Kanten

(d) **Reguläre Matroide und totale Unimodularität**
Da jedes reguläre Matroid M binär ist, hat es auch eine binäre Standardrepräsentation (I, B). Und hier besteht eine Verbindung zu Kapitel 13 aus ADM I. Ist (I, B) eine binäre Standardrepräsentation eines binären Matroids M, so kann man beweisen, dass M regulär genau dann ist, wenn die „binären Einsen“, also die Werte $b_{ij} = 1$ von B, so „umsigniert“ (d. h. durch eine der beiden rationalen Zahlen $+1$ oder -1 ersetzt) werden können, dass die entstehende Matrix B' total unimodular (als Matrix über $\mathbb{Q}$ betrachtet) ist. △

Es folgt nun eine Liste weiterer interessanter Matroide.

(2.9) Beispiel.

(a) **Cographische Matroide**
Gegeben sei ein Graph $G = (V, E)$. Ein *Cokreis* ist eine Kantenmenge, deren Entfernung aus G die Komponentenzahl erhöht und die (mengeninklusionsweise) minimal bezüglich dieser Eigenschaft ist. Ein *Schnitt* ist eine Kantenmenge der Form

$$\delta(W) = \{ij \in E \mid i \in W, j \in V \setminus W\}, \quad W \subseteq V.$$

Jeder Cokreis ist offenbar ein Schnitt, und es ist einfach einzusehen, dass die Cokreise gerade die minimalen nicht-leeren Schnitte von G sind. Sind $\delta(W)$ und $\delta(W')$ verschiedene Cokreise und ist die Kante ij in beiden enthalten, so ist $\delta(W \triangle W')$ ein Schnitt, der ij nicht enthält. (Hier bezeichnet $W \triangle W'$ die *symmetrische Differenz* $(W \cup W') \setminus (W \cap W')$.) Ist $\delta(W \triangle W')$ kein Cokreis, so enthält er einen Cokreis, der natürlich ij auch nicht enthält. Daraus folgt, dass die Antikette der Cokreise das Axiom (C.1) erfüllt und somit das Zirkuitsystem eines Matroids auf E ist. Ein Matroid, das isomorph zu einem so definierten Matroid ist, heißt *cographisch*. Ist $G = (V, E)$ ein zusammenhängender Graph und M das cographische Matroid auf G, so ist das Basissystem $\mathcal{B}$ von M gegeben durch

$$\mathcal{B} = \{B \subseteq E \mid \exists \text{ aufspannender Baum } T \subseteq E \text{ mit } B = E \setminus T\}.$$

(b) **Uniforme Matroide**
Sei E eine Menge mit n Elementen, dann ist die Menge aller Teilmengen von E mit höchstens k Elementen ein Matroid auf E. Dieses Matroid heißt *uniform* und ist durch die Angabe von k und n bis auf Isomorphie eindeutig bestimmt. Dieses Matroid wird mit $U_{k,n}$ bezeichnet. Das Basissystem von $U_{k,n}$ wird durch die Menge der Teilmengen von E mit genau k Elementen gebildet. Das Zirkuitsystem von $U_{k,n}$ besteht aus den Teilmengen von E mit $k+1$ Elementen. Die Matroide $U_{n,n}$ (d. h. die Matroide, in denen alle Mengen unabhängig sind) heißen *frei* (oder *trivial*). Für freie Matroide gilt $\mathcal{C} = \emptyset$, $\mathcal{B} = \{E\}$, $r(F) = |F|$ für alle $F \subseteq E$. Das uniforme Matroid $U_{2,4}$ ist das kleinste nicht binäre Matroid. Überlegen Sie sich einen Beweis hierfür!

(c) **Partitionsmatroide**
Sei E eine endliche Menge, und $E_1, \ldots, E_k$ seien nicht-leere Teilmengen von E mit $E_i \cap E_j = \emptyset$, $i \neq j$, und $\bigcup_{i=1}^{k} E_i = E$. Seien $b_1, \ldots, b_k$ nicht-negative ganze Zahlen, dann ist $\mathcal{I} := \{I \subseteq E \mid |I \cap E_i| \leq b_i, i = 1, \ldots, k\}$ ein Matroid auf E, genannt *Partitionsmatroid*.

(d) **Transversalmatroide**
Sei E eine endliche Menge, $(E_i)_{i \in I}$ sei eine endliche Familie von Teilmengen von E. Eine Teilmenge $T \subseteq E$ ist eine *teilweise Transversale* (oder partielles Repräsentantensystem) von $(E_i)_{i \in I}$, falls es eine Indexmenge $J \subseteq I$ gibt mit $|J| = |T|$ und eine Bijektion $\pi : T \to J$, so dass $t \in E_{\pi(t)}$ für alle $t \in T$. Die Menge aller teilweisen Transversalen ist das Unabhängigkeitssystem eines Matroids auf E. (Dieses Ergebnis ist nicht trivial und hatte einen wesentlichen Einfluss auf die Transversaltheorie.) $\triangle$

Wir wollen nun noch einige konkrete Beispiele von Matroiden angeben und ihre Basis-, Zirkuitsysteme etc. explizit auflisten.

Betrachten wir den folgenden, in Abbildung 2.2 dargestellten Graphen $G = (V, E)$ mit $E = \{1, 2, \ldots, 13\}$. Das Zirkuitsystem $\mathcal{C}$ des graphischen Matroids auf E ist gegeben durch die Menge aller Kreise in G, d. h.

$$\mathcal{C} = \{\{1, 2, 3\}, \{4, 5, 6, 7\}, \{9, 10, 11\}, \{11, 12, 13\}, \{9, 10, 12, 13\}\}.$$

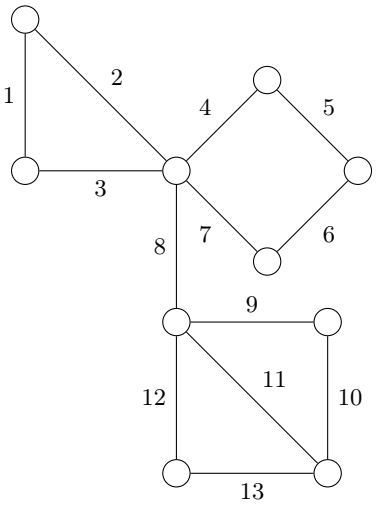


Abbildung 2.2: Graph G mit 13 Kanten

Das Zirkuitsystem $\mathcal{C}^*$ des cographischen Matroids auf E ist gegeben durch die Menge aller minimalen Schnitte, d. h.

$$\begin{aligned}\mathcal{C}^* = \; & \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{4, 5\}, \{4, 6\}, \{4, 7\}, \{5, 6\}, \{5, 7\}, \{6, 7\}, \{8\}, \\ & \{9, 10\}, \{9, 11, 12\}, \{9, 11, 13\}, \{10, 11, 12\}, \{10, 11, 13\}, \{12, 13\}\}.\end{aligned}$$

Das Basissystem $\mathcal{B}$ des graphischen Matroids des folgenden Graphen $G = (V, E)$ (siehe Abbildung 2.3) mit $E = \{1, 2, 3, 4\}$ ist gegeben durch

$$\mathcal{B} = \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}\},$$

und das Basissystem $\mathcal{B}^*$ des cographischen Matroids bezüglich G ist die folgende Antikette:

$$\mathcal{B}^* = \{\{4\}, \{3\}, \{2\}\}.$$

Das graphische Matroid des in Abbildung 2.4 dargestellten Graphen ist das uniforme Matroid $U_{2,3}$. Uniforme Matroide können also auch isomorph zu graphischen sein. Das uniforme Matroid $U_{2,4}$ ist jedoch nicht graphisch (Übungsaufgabe).

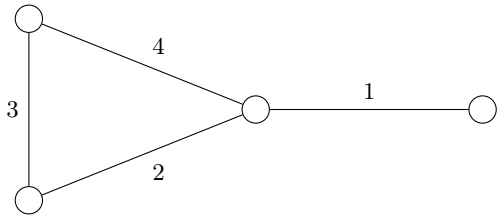


Abbildung 2.3: Graph G mit 4 Kanten

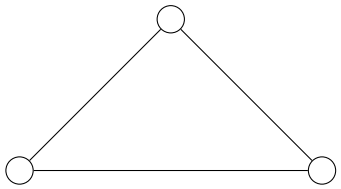


Abbildung 2.4: Graph zum uniformen Matroid $U_{2,3}$

Betrachten wir die Matrix

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

als Matrix über dem Körper $\mathbb{R}$ oder $\mathbb{Q}$. Das Zirkuitsystem $\mathcal{C}$ des Matrix-Matroids M bezüglich A ist offenbar gegeben durch

$$\mathcal{C} = \{\{1,2,3\},\{2,3,4\},\{1,4\}\},$$

und das Basissystem $\mathcal{B}$ des Matrix-Matroids M ist

$$\mathcal{B} = \{\{1,2\},\{1,3\},\{2,3\},\{2,4\},\{3,4\}\}.$$

Ein aus der endlichen Geometrie stammendes Beispiel ist das **Fano-Matroid**, das häufig mit dem Symbol F_7 bezeichnet wird. Betrachten Sie Abbildung 2.5. Dies ist eine graphische Darstellung der Fano-Ebene. Die Fano-Ebene hat 7 Punkte und 7 Geraden. Die 7 Geraden werden durch die 6 geraden Linien $\{1,2,6\}$, …, $\{3,6,7\}$ und den Kreis, der durch die Punkte $\{4,5,6\}$ geht, repräsentiert. Das Fano-Matroid F_7 ist auf der Menge $E = \{1,\ldots,7\}$ definiert. Eine Teilmenge B von E ist genau dann eine Basis von F_7, wenn $|B| = 3$ und wenn die drei zu B gehörigen Punkte nicht kollinear sind, also nicht auf einer Geraden liegen. Die Menge $\{1,2,3\}$ ist somit eine Basis, $\{4,5,6\}$ dagegen nicht. Das Fano-Matroid ist binär. Wählen wir die Basis $T = \{1,2,3\}$, so ergibt die in (2.8)(c) beschriebene Konstruktion der Standardrepräsentation die folgende Matrix:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

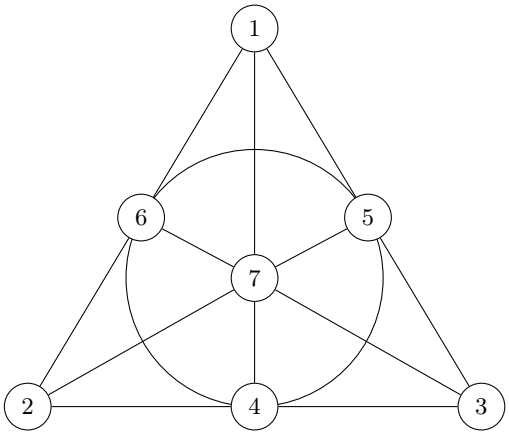


Abbildung 2.5: Fano-Ebene

die das Fano-Matroid F_7 über $GF(2)$ repräsentiert. Das Fano-Matroid F_7 ist nicht regulär. Ohne näher darauf einzugehen, sei ein tiefliegender Satz von Tutte zitiert:

> *Ein binäres Matroid ist regulär genau dann, wenn es weder F_7 noch das zu F_7 duale Matroid als Minor enthält.*

Wir wollen nun noch einen einfachen, aber interessanten Zusammenhang zwischen Unabhängigkeitssystemen und Matroiden erwähnen.

(2.10) Satz. *Jedes Unabhängigkeitssystem ist als Durchschnitt von Matroiden darstellbar, d. h. ist $\mathcal{I}$ ein Unabhängigkeitssystem auf E, dann gibt es Matroide $\mathcal{I}_1, \ldots, \mathcal{I}_k$ auf E mit*

$$\mathcal{I} = \bigcap_{i=1}^{k} \mathcal{I}_i.$$

△

Beweis. Sei $\mathcal{C}$ das zu $\mathcal{I}$ gehörige Zirkuitsystem. Jedes Zirkuit $C \in \mathcal{C}$ definiert eine Antikette $\{C\} \subseteq 2^E$, die trivialerweise das Axiom (C.1) aus (2.7) erfüllt. Also ist das zu dem Zirkuitsystem $\{C\}$ gehörige Unabhängigkeitssystem $\mathcal{I}_C$ ein Matroid. Wir behaupten nun

$$\mathcal{I} = \bigcap_{C \in \mathcal{C}} \mathcal{I}_C.$$

Ist $I \in \mathcal{I}$, so ist kein Zirkuit $C \in \mathcal{C}$ in I enthalten, folglich ist nach (2.2) $I \in \mathcal{I}_C$ für alle $C \in \mathcal{C}$. Sei umgekehrt $I \in \mathcal{I}_C$ für alle $C \in \mathcal{C}$, so heißt dies, dass kein Zirkuit $C \in \mathcal{C}$ in I enthalten ist, und somit, dass I ein Element von $\mathcal{I}$ ist. □

Die im Beweis von Satz (2.10) angegebene Konstruktion zur Darstellung eines Unabhängigkeitssystems als Durchschnitt von Matroiden produziert i. A. eine riesige Zahl von Matroiden, die das Gewünschte leisten. Häufig kommt man mit viel weniger Matroiden aus.

Betrachten wir z. B. die Menge der Branchings $\mathcal{I} \subseteq 2^A$ in einem Digraphen $D = (V, A)$. Man kann einfach zeigen, dass das zugehörige Zirkuitsystem $\mathcal{C}$ aus den inklusionsminimalen Mengen der Vereinigung der folgenden Antiketten $\mathcal{C}_1$ und $\mathcal{C}_2$ besteht:

$\mathcal{C}_1 := \{C \subseteq A \mid |C| = 2$ und die Endknoten der beiden Bögen in C sind identisch$\}$,
$\mathcal{C}_2 := \{C \subseteq A \mid C$ ist ein Kreis (die Bogenrichtungen spielen keine Rolle)$\}$.

$\mathcal{C}_1$ ist das Zirkuitsystem eines Partitionsmatroids auf A, dessen Unabhängigkeitssystem gegeben ist durch $\{B \subseteq A \mid |B \cap \delta^-(v)| \leq 1 \ \forall v \in V\}$, und $\mathcal{C}_2$ ist das Zirkuitsystem des graphischen Matroids auf D (hierbei wird D als Graph aufgefasst, d. h. die Bogenrichtungen werden ignoriert). Daraus folgt, dass das Unabhängigkeitssystem der Branchings Durchschnitt von 2 Matroiden ist. Für den vollständigen Digraphen D mit n Knoten hätte die Konstruktion im Beweis von Satz (2.10) insgesamt

$$n\binom{n-1}{2} + \sum_{k=2}^{n} \binom{n}{k}(k-1)!$$

verschiedene Matroide geliefert.

Das Unabhängigkeitssystem $\tilde{\mathcal{T}}$ der Teilmengen von Touren im vollständigen Digraphen D_n mit n Knoten, siehe (2.5)(d), ist als Durchschnitt von 3 Matroiden darstellbar (Übungsaufgabe). Also ist das asymmetrische Travelling-Salesman-Problem als Optimierungsproblem über dem Durchschnitt von 3 Matroiden formulierbar.

2.3 Orakel

Um Eigenschaften von Matroiden oder Unabhängigkeitssystemen überprüfen zu können, muss man sich natürlich fragen, wie man Matroide geeignet darstellt, um z. B. ein Computerprogramm schreiben zu können, das Matroide als Input akzeptiert.

Betrachten wir zum Beispiel das in (2.3) formulierte Optimierungsproblem $\max\{c(I) \mid I \in \mathcal{I}\}$ über einem Unabhängigkeitssystem $\mathcal{I}$ auf E (Spezialfall: $(E, \mathcal{I})$ ist ein Matroid). Ist $\mathcal{I}$ als Liste aller unabhängigen Mengen gegeben, so ist das Optimierungsproblem völlig trivial. Wir durchlaufen die Liste, rechnen für jede Menge I der Liste den Wert $c(I)$ aus und wählen eine Menge I^*, so dass $c(I^*)$ maximal ist. Die Laufzeit dieses Enumerationsalgorithmus ist linear in der Inputlänge des Unabhängigkeitssystems.

Viele der Matroide und Unabhängigkeitssysteme, die wir in den voraufgegangenen Abschnitten definiert haben, sind jedoch in wesentlich kompakterer Form gegeben. Zum Beispiel ist ein Matrix-Matroid (2.8)(b) durch eine Matrix A (mit der Information, dass $I \subseteq E$ unabhängig genau dann ist, wenn die Spalten $A_{\cdot i}$, $i \in I$, linear unabhängig sind) gegeben, ein graphisches Matroid (2.8)(a) durch einen Graphen $G = (V, E)$ (zusammen mit der Information, dass $I \subseteq E$ unabhängig ist genau dann, wenn I keinen Kreis enthält), ein cographisches Matroid (2.9)(a) durch einen Graphen $G = (V, E)$ (zusammen mit der Information, dass $C \subseteq E$ ein Zirkuit ist genau dann, wenn C ein Cokreis ist). Würden wir die Inputlänge des Matroids als die Länge der Kodierung der Matrix A (für (2.8)(b)) bzw. als die Länge der Kodierung des Graphen G (für (2.8)(a)) definieren, hätte

unser trivialer Enumerationsalgorithmus exponentielle Laufzeit in der Kodierungslänge dieses kompakten Inputs.

Die Frage ist also: Was ist eine „geeignete“ Kodierung eines Matroids? Motiviert durch die gerade angegebenen Beispiele könnte man glauben, dass die Angabe einer Liste aller unabhängigen Mengen eine unökonomische Methode sei. Jedoch geht es im allgemeinen nicht viel besser. Man kann nämlich zeigen, dass es mindestens

$$2^{2^{n/2}} \text{ Matroide mit } n \text{ Elementen}$$

gibt. Daraus folgt, dass es bei jeder beliebigen Kodierungsart immer Matroide gibt, deren Kodierung eine Länge hat, die mindestens $2^{n/2}$ beträgt.

Aufgrund dieser Tatsache hat sich ein anderes Konzept der Repräsentation von Matroiden durchgesetzt und als außerordentlich nützlich erwiesen. Matroide werden durch „Orakel“ dargestellt.

Wir treffen folgende Definition. Die *Kodierungslänge eines Matroids* $M = (E, \mathcal{I})$ ist die Anzahl der Elemente von E. Das Matroid selbst ist in einem Orakel „versteckt“, wobei das Orakel als ein Computerprogramm (Subroutine) interpretiert werden kann, das Fragen eines speziellen Typs beantwortet. Wir geben einige Beispiele (diese gelten natürlich nicht nur für Matroide, sondern analog auch für Unabhängigkeitssysteme) und gehen davon aus, dass wir die Grundmenge E kennen.

(2.11) Beispiel.

(a) **Unabhängigkeitsorakel**
Für jede Menge $F \subseteq E$ können wir das Orakel fragen, ob F unabhängig ist. Das Orakel antwortet mit „ja“ oder „nein“.

(b) **Zirkuitorakel**
Für jede Menge $F \subseteq E$ können wir das Orakel fragen, ob F ein Zirkuit ist. Das Orakel antwortet mit „ja“ oder „nein“.

(c) **Basisorakel**
Für jede Menge $F \subseteq E$ können wir das Orakel fragen, ob F eine Basis von E ist. Das Orakel antwortet mit „ja“ oder „nein“.

(d) **Rangorakel**
Für jede Menge $F \subseteq E$ können wir das Orakel fragen, wie groß der Rang von F ist. Die Antwort des Orakels ist „$r(F)$“. △

Man sieht sofort, dass dieses theoretische Orakelkonzept dem Konzept von Unterprogrammen der Programmierung entspricht.

Wenn wir also Algorithmen betrachten wollen, die Eigenschaften von Matroiden überprüfen, so gehen wir immer davon aus, dass ein Matroid (oder Unabhängigkeitssystem) durch die Grundmenge E und ein Orakel gegeben ist, das im Verlaufe des Algorithmus befragt werden kann. Bei der Komplexitätsanalyse von Algorithmen für Matroide (Unabhängigkeitssysteme) wird dann ein Orakelaufruf (Unterprogrammaufruf) als ein Schritt

gezählt. Achtung! Die Laufzeit beim Lesen der Antwort wird wie üblich berechnet. Gibt also ein Orakel eine Antwort, die z. B. $2^{|E|}$ Speicherplätze benötigt, so hat der Algorithmus automatisch eine exponentielle Laufzeit. Bei den in (2.11) angegebenen Orakeln kommt so ein Fall allerdings nie vor! Man nennt Verfahren, die Orakel aufrufen können, *Orakelalgorithmen*. Ist ihre Laufzeit in dem oben angegebenen Sinne polynomial in $|E|$, so sagt man, dass die Verfahren *orakelpolynomial* sind.

Hat man einen Algorithmus, dessen Laufzeit orakelpolynomial in $|E|$ ist, so ist der Algorithmus für alle die Matroide (Unabhängigkeitssysteme) im üblichen Sinne polynomial, für die das Orakel durch einen in $|E|$ polynomialen Algorithmus realisiert werden kann. Dies ist zum Beispiel für Matrix-Matroide der Fall. Ein Unabhängigkeitsorakel kann man bei einer gegebenen Matrix durch Rangbestimmung mit Gauß-Elimination realisieren (analog die drei anderen Orakel). Dies gilt auch für graphische und cographische Matroide, die durch einen Graphen $G = (V, E)$ gegeben sind. Zum Beispiel kann man die Unabhängigkeit einer Menge F im graphischen Matroid (F enthält keinen Kreis) durch Depth-First-Search testen; auch die übrigen drei Orakel kann man durch Algorithmen realisieren, deren Laufzeit polynomial in $|E|$ ist.

Eine wichtige Frage ergibt sich sofort. Sind die verschiedenen Orakel algorithmisch „äquivalent“? Das heißt, kann man ein Orakel durch ein anderes Orakel in orakelpolynomialer Zeit simulieren und umgekehrt? Zum Beispiel: Ist ein Unabhängigkeitsorakel gegeben, kann man dann einen Algorithmus entwerfen (der nur dieses Orakel benutzt und orakelpolynomial ist), der für jede Menge $F \subseteq E$ korrekt entscheidet, ob F ein Zirkuit ist oder nicht?

Für Matroide und die vier oben angegebenen Orakel wurde diese Frage von Hausmann und Korte (1981) wie folgt beantwortet.

(2.12) Satz. *Sei $M = (E, \mathcal{I})$ ein beliebiges Matroid.*

(a) *Das Unabhängigkeitsorakel und das Rangorakel sind bezüglich M „äquivalent“.*

(b) *Das Basisorakel und das Zirkuitorakel können durch das Unabhängigkeitsorakel sowie durch das Rangorakel in orakelpolynomialer Zeit simuliert werden.*

(c) *Es gibt keine anderen orakelpolynomialen Beziehungen zwischen diesen Orakeln.* △

Man kann diesen Satz graphisch durch Abbildung 2.6 veranschaulichen. Ein Pfeil in diesem Diagramm besagt, dass das Orakel am Ende des Pfeils durch polynomial viele Aufrufe des Orakels an der Spitze des Pfeils simuliert werden kann. Ist zwischen zwei Kästchen kein Pfeil (in irgendeiner der beiden Richtungen), so besagt dies auch, dass es keine orakelpolynomiale Simulation dieser Art gibt. Z. B. kann man das Basisorakel nicht durch das Zirkuitorakel in orakelpolynomialer Zeit simulieren und umgekehrt. Dies zeigt man dadurch, dass man eine Klasse von Beispielen angibt, bei denen zur Entscheidung der Frage, ob eine Menge I eine Basis (ein Zirkuit) ist, eine Anzahl von Aufrufen des Zirkuitorakels (Basisorakels) notwendig ist, die exponentiell in $|E|$ ist.

Hausmann und Korte (1980) haben dieselbe Fragestellung auch für allgemeine Unabhängigkeitssysteme untersucht, die – wie wir in Abschnitt 2.1 gesehen haben – äquivalent

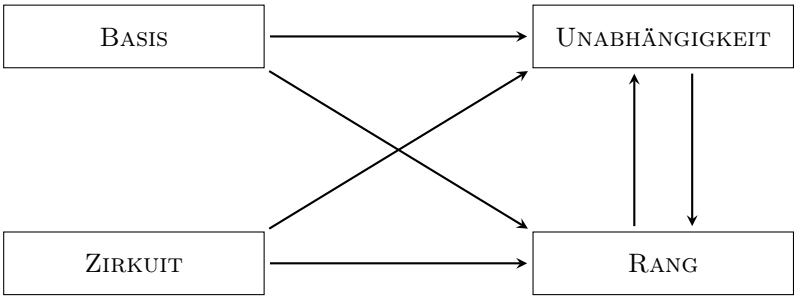


Abbildung 2.6: Orakelpolynomiale Reduktionen für Matroide

durch Zirkuitsysteme, Basissysteme oder Rangfunktionen gegeben werden können. Der entsprechende Satz ist bildlich in Abbildung 2.7 veranschaulicht (Interpretation wie bei Abbildung 2.6).

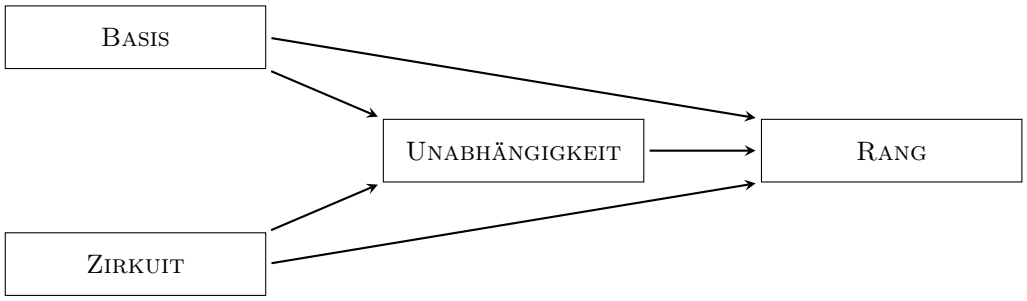


Abbildung 2.7: Orakelpolynomiale Reduktionen für allgemeine Unabhängigkeitssysteme

Insbesondere folgt, dass bei Unabhängigkeitssystemen das Rangorakel das stärkste ist. Durch dieses lassen sich die übrigen Orakel in orakelpolynomialer Zeit simulieren. Jedoch sind hier keine zwei Orakel algorithmisch „äquivalent“ (sie sind natürlich logisch äquivalent).

2.4 Optimierung über Unabhängigkeitssystemen

Wir wollen nun die Frage untersuchen, ob bzw. wie gut ein Optimierungsproblem der Form (2.3)

$$\max\{c(I) \mid I \in \mathcal{I}\},$$

wobei $\mathcal{I}$ ein Unabhängigkeitssystem auf einer Grundmenge E ist, gelöst werden kann. Wir betrachten dazu den folgenden trivialen Algorithmus, vergleiche ADM I, (5.7).

(2.13) Algorithmus Greedy-max für Unabhängigkeitssysteme.

Eingabe: Grundmenge $E = \{1, \ldots, n\}$ mit Gewichten $c_i \in \mathbb{R}$ für alle $i \in E$. Ferner ist ein Unabhängigkeitssystem $\mathcal{I} \subseteq 2^E$ durch ein Unabhängigkeitsorakel (siehe (2.11)(a)) gegeben.

Ausgabe: Eine unabhängige Menge $I_g \in \mathcal{I}$.

1. Sortiere die Gewichte in nicht aufsteigender Reihenfolge (d. h. nach Beendigung von Schritt 1 können wir annehmen, dass $c_1 \geq c_2 \geq \cdots \geq c_n$ gilt).
2. Setze $I := \emptyset$.
3. FOR $i = 1$ TO n DO:

   Ist $c_i \leq 0$, gehe zu 4.

   Ist $I \cup \{i\}$ unabhängig (Orakelaufruf), dann setze $I := I \cup \{i\}$.
4. Setze $I_g := I$ und gib I_g aus. △

Der Greedy-Algorithmus durchläuft (nach der Sortierung in Schritt 1) die Elemente von E genau einmal, entweder er nimmt ein Element in die Menge I auf, oder er verwirft es für immer. Die am Ende des Verfahrens gefundene Lösung I_g nennen wir *Greedy-Lösung.*

Für eine Menge $F \subseteq E$ setzen wir

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

und nennen $r_u(F)$ den *unteren Rang von* F. Wir setzen

$$q := \min_{F \subseteq E, r(F)>0} \frac{r_u(F)}{r(F)}$$

und nennen q den *Rangquotient* von $\mathcal{I}$. Ist $(E, \mathcal{I})$ ein Matroid, so gilt nach (I.3″) natürlich $r_u(F) = r(F)$ für alle $F \subseteq E$ und somit $q = 1$. Das folgende Resultat wurde von Jenkyns (1976) bewiesen.

(2.14) Satz. *Sei $\mathcal{I}$ ein Unabhängigkeitssystem auf E, seien I_g eine Greedy-Lösung von (2.13) und I_0 eine optimale Lösung von* (2.3)*, dann gilt*

$$q \leq \frac{c(I_g)}{c(I_0)} \leq 1,$$

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

Beweis. Wir können o. B. d. A. annehmen, dass $c_i > 0$ für $i = 1, \ldots, n$ und dass $c_1 \geq c_2 \geq \cdots \geq c_n$ gilt. Aus notationstechnischen Gründen führen wir ein Element $n + 1$ ein mit $c_{n+1} = 0$. Wir setzen

$$E_i := \{1, \ldots, i\}, \quad i = 1, \ldots, n.$$

Es gilt dann offenbar:

$$c(I_g) = \sum_{i=1}^{n} |I_g \cap E_i|(c_i - c_{i+1}),$$

$$c(I_0) = \sum_{i=1}^{n} |I_0 \cap E_i|(c_i - c_{i+1}).$$

Da $I_0 \cap E_i \subseteq I_0$, gilt $I_0 \cap E_i \in \mathcal{I}$, und somit $|I_0 \cap E_i| \leq r(E_i)$. Die Vorgehensweise des Greedy-Algorithmus impliziert, dass $I_g \cap E_i$ eine Basis von E_i ist, also dass $|I_g \cap E_i| \geq r_u(E_i)$ gilt. Daraus folgt

$$|I_g \cap E_i| \;\geq\; |I_0 \cap E_i| \, \frac{r_u(E_i)}{r(E_i)} \;\geq\; |I_0 \cap E_i| \, q, \quad i = 1, \ldots, n$$

und somit

$$\begin{aligned} c(I_g) &= \sum_{i=1}^{n} |I_g \cap E_i|(c_i - c_{i+1}) \\ &\geq \sum_{i=1}^{n} |I_0 \cap E_i| q (c_i - c_{i+1}) \\ &= q \sum_{i=1}^{n} |I_0 \cap E_i|(c_i - c_{i+1}) \\ &= c(I_0) q. \end{aligned}$$

Die Ungleichung $1 \geq \frac{c(I_g)}{c(I_0)}$ ist trivial. Damit haben wir die Gültigkeit der Ungleichungskette bewiesen.

Sei nun $F \subseteq E$ mit $q = \frac{r_u(F)}{r(F)}$. Wir können annehmen, dass $F = \{1, \ldots, k\}$ gilt und dass $B = \{1, \ldots, p\} \subseteq F$ eine Basis von F ist mit $|B| = r_u(F)$. Wir setzen $c_i = 1$, $i = 1, \ldots, k$, und $c_i = 0$, $i = k+1, \ldots, n$. Dann liefert der Greedy-Algorithmus die Greedy-Lösung $I_g = B$ mit $c(I_g) = r_u(F)$, während für jede Optimallösung I_0 gilt $c(I_0) = r(F)$. Also wird für diese spezielle 0/1-Zielfunktion die linke Ungleichung in der Aussage des Satzes mit Gleichheit angenommen. □

(2.15) Korollar. *Sei $\mathcal{I}$ ein Unabhängigkeitssystem auf E. Dann sind äquivalent:*

(a) *$(E, \mathcal{I})$ ist ein Matroid.*

(b) *Für alle Gewichte $c \in \mathbb{R}^E$ liefert der Greedy-Algorithmus (2.13) eine Optimallösung von* (2.3).

(c) *Für alle Gewichte mit* 0/1*-Koeffizienten liefert der Greedy-Algorithmus (2.13) eine Optimallösung von* (2.3). △

(2.15) wurde von verschiedenen Autoren unabhängig voneinander bewiesen. Edmonds (1971) zeigte insbesondere, wie man den Greedy-Algorithmus für Matroide als ein Verfahren zur Lösung spezieller linearer Programme deuten kann. Er liefert in der Tat auch duale Lösungen. Wir gehen darauf in 3.2 ein.

Da der Algorithmus (5.7) GREEDY-MAX aus ADM I offenbar eine Spezialisierung des Greedy-Algorithmus (2.13) für das graphische Matroid ist, liefert Folgerung (2.15) einen weiteren Beweis von ADM I, Satz (5.8).

Satz (2.14) ist ein Prototyp von Abschätzungssätzen für die Qualität von heuristischen Lösungen, wie wir sie später noch mehrfach kennenlernen werden. Der Greedy-Algorithmus (2.13) ist offenbar orakelpolynomial. Immer dann, wenn wir den Orakelaufruf (Unabhängigkeitstest) in Schritt 3 durch einen polynomialen Algorithmus realisieren können, ist der Greedy-Algorithmus ein polynomialer Algorithmus (im üblichen Sinne). Eine solche polynomiale Realisierung ist z. B. auf triviale Weise für das Cliquenproblem, Stabile-Menge-Problem, Travelling-Salesman-Problem und das azyklische Subdigraphenproblem möglich. Würde der Greedy-Algorithmus auch in diesen Fällen immer Optimallösungen liefern, hätten wir einen Beweis für $\mathcal{P} = \mathcal{NP}$ gefunden, da alle gerade aufgelisteten Probleme $\mathcal{NP}$-schwer sind. Wir können also nicht erwarten, dass der Greedy-Algorithmus immer optimale Antworten produziert. Die Frage, wie schlecht im schlechtest möglichen Falle eine Greedy-Lösung ist, beantwortet Satz (2.14) auf bestmögliche Weise. Er gibt eine sogenannte *Gütegarantie* an, die besagt, dass der Wert $c(I_g)$ jeder Greedy-Lösung nicht schlechter ist als $qc(I_o)$, wobei q der bereits erwähnte Rangquotient ist. Der Rangquotient liefert also eine Gütegarantie für die Lösungsqualität des Greedy-Algorithmus. Der Rangquotient ist im allgemeinen nicht einfach auszurechnen. Eine Abschätzung wird gegeben durch:

(2.16) Satz. *Sei $\mathcal{I}$ ein Unabhängigkeitssystem auf E, und $(E, \mathcal{I}_i)$, $i = 1, \ldots, k$ sei eine minimale Zahl von Matroiden mit $\mathcal{I} = \bigcap_{i=1}^{k} \mathcal{I}_i$, dann gilt*

$$\min_{F \subseteq E, r(F) > 0} \frac{r_u(F)}{r(F)} \geq \frac{1}{k}. \qquad \triangle$$

Daraus folgt zum Beispiel: Ist $\mathcal{I}$ ein Unabhängigkeitssystem, das Durchschnitt von 2 Matroiden ist, dann ist der Wert der Greedy-Lösung mindestens halb so groß wie der Wert der Optimallösung von (2.3). Insbesondere liefert also der Greedy-Algorithmus für Branchingprobleme Lösungen, deren Wert mindestens die Hälfte des Optimums beträgt.

Für das Branching-Problem gibt es einen polynomialen Lösungsalgorithmus. Es gibt also offenbar auch Probleme über Unabhängigkeitssystemen, die keine Matroide sind und für die effiziente Optimierungsverfahren existieren. Ein sehr tiefliegendes Resultat ist der folgende von Edmonds (1979) und Lawler (1975) gefundene Satz.

(2.17) Satz. *Seien $(E, \mathcal{I}_1)$ und $(E, \mathcal{I}_2)$ zwei Matroide gegeben durch Unabhängigkeitsorakel, dann gibt es einen Algorithmus, der für jede beliebige Zielfunktion c das Problem $\max\{c(I) \mid I \in \mathcal{I}_1 \cap \mathcal{I}_2\}$ in orakelpolynomialer Zeit löst.* $\triangle$

Der Algorithmus, der den Beweis des obigen Satzes liefert, ist relativ kompliziert, und sein Korrektheitsbeweis benötigt Hilfsmittel aus der Matroidtheorie, die uns hier nicht zur Verfügung stehen. Deshalb verzichten wir auf eine Angabe und Analyse dieses (sehr interessanten) Verfahrens.

Man fragt sich nun natürlich sofort, ob auch über dem Durchschnitt von 3 Matroiden in (orakel-)polynomialer Zeit optimiert werden kann. Dies geht (vermutlich) i. A. nicht. Man kann zeigen, dass das Optimierungsproblem für Unabhängigkeitssysteme, die Durchschnitte von 3 Matroiden sind, Probleme enthält, die $\mathcal{NP}$-schwer sind.

Ein Beispiel hierfür ist das asymmetrische Travelling-Salesman-Problem (ATSP). Das zugehörige Unabhängigkeitssystem $\mathcal{H}$ ist wie folgt definiert:

$$\mathcal{H} := \{S \subseteq A_n \mid S \text{ ist Teilmenge eines gerichteten hamiltonschen Kreises in } D_n = (V, A_n)\},$$

wobei D_n der vollständige gerichtete Graph auf n Knoten ist. Wir modifizieren D_n zu dem Graphen $D'_n = (V', A'_n)$, indem wir den Knoten $1 \in V$ in zwei neue Knoten $1, n+1 \in V'$ aufspalten. Alle Bögen aus A_n, die den Knoten 1 aus D_n als Anfangsknoten haben, haben auch $1 \in V'$ als Anfangsknoten; die Bögen aus A_n, die $1 \in V$ als Endknoten haben, bekommen den neuen Knoten $n+1 \in V'$ als Endknoten zugewiesen. Diese Konstruktion bewirkt, dass jedem gerichteten hamiltonschen Kreis in D_n ein gerichteter hamiltonscher Weg von 1 nach $n+1$ in D'_n entspricht und umgekehrt. Das ATSP-Unabhängigkeitssystem $\mathcal{H}$ ist dann (bis auf Benamung einiger Knoten und Bögen) identisch mit

$$\mathcal{H}' := \{S \subseteq A'_n \mid S \text{ ist Teilmenge eines gerichteten hamiltonschen Wegs von 1 nach } n+1 \text{ in } D'_n\}.$$

Definieren wir auf A'_n die drei folgenden Unabhängigkeitssysteme

$$\begin{aligned}\mathcal{W} &:= \{W \subseteq A'_n \mid W \text{ ist Wald }\},\\ \mathcal{P}^- &:= \{F \subseteq A'_n \mid |F \cap \delta^-(v)| \leq 1 \; \forall v \in V'\},\\ \mathcal{P}^+ &:= \{F \subseteq A'_n \mid |F \cap \delta^+(v)| \leq 1 \; \forall v \in V'\},\end{aligned}$$

so sind $\mathcal{W}$, $\mathcal{P}^-$, $\mathcal{P}^+$ Unabhängigkeitssysteme von Matroiden auf A'_n, und es gilt $\mathcal{H}' = \mathcal{W} \cap \mathcal{P}^- \cap \mathcal{P}^+$.

Wir wollen uns nun noch kurz dem Optimierungsproblem (2.4) über Basissystemen zuwenden. Wie bei Bäumen und Wäldern vorgeführt (siehe ADM I, Abschnitt 5.2), kann man ein Maximierungsproblem des Typs (2.3) mit Hilfe einer polynomialen Transformation in ein Minimierungsproblem (2.4) überführen und umgekehrt. Analog lässt sich auch aus (2.13) ein Minimierungsalgorithmus ableiten.

(2.18) Algorithmus Greedy-min für Basissysteme.

Eingabe: Grundmenge $E = \{1, \ldots, n\}$ mit Gewichten $c_i \in \mathbb{R}$ für alle $i \in E$, ein Unabhängigkeitssystem $\mathcal{I} \subseteq 2^E$ gegeben durch ein Unabhängigkeitsorakel.

Ausgabe: Eine Basis $B_g \in \mathcal{I}$.

1. Sortiere die Gewichte in nicht absteigender Reihenfolge (so dass nach Beendigung von Schritt 1 gilt: $c_1 \leq c_2 \leq \cdots \leq c_n$).
2. Setze $I := \emptyset$.
3. FOR $i = 1$ TO n DO:

   Ist $I \cup \{i\} \in \mathcal{I}$ (Orakelaufruf), dann setze $I := I \cup \{i\}$.
4. Setze $B_g := I$ und gib B_g aus. △

Offenbar ist B_g eine Basis der Grundmenge E des Unabhängigkeitssystems $\mathcal{I}$, gehört also zum zugehörigen Basissystem $\mathcal{B}$. Folgerung (2.15) impliziert:

(2.19) Satz. *Sei $\mathcal{I}$ ein Unabhängigkeitssystem auf E und $\mathcal{B}$ das zugehörige Basissystem. Dann sind äquivalent:*

(a) *$(E, \mathcal{I})$ ist ein Matroid.*

(b) *Für alle Gewichte $c \in \mathbb{R}^E$ liefert der Greedy-Algorithmus (2.18) eine Optimallösung von $\min\{c(B) \mid B \in \mathcal{B}\}$.*

(c) *Für alle Gewichte $c \in \{0,1\}^E$ liefert der Greedy-Algorithmus (2.18) eine Optimallösung von $\min\{c(B) \mid B \in \mathcal{B}\}$.* △

Sortiert man in Schritt 1 von (2.18) die Gewichte in nicht aufsteigender Reihenfolge, so erhält man offenbar eine Basis maximalen Gewichts. Satz (2.19) gilt analog, wenn man „min“ durch „max“ ersetzt.

Leider kann man die schöne Abschätzung aus Satz (2.14) für die Gütegarantie von (2.13) nicht ohne weiteres auf Algorithmus (2.18) übertragen. In der Tat gibt es keine universelle Konstante, die bezüglich eines Problems die Qualität des Ergebnisses von (2.18) unabhängig von den Zielfunktionskoeffizienten beschränkt. Betrachten wir z. B. das Cliquen-Problem auf dem in Abbildung 2.8 dargestellten Graphen mit der Gewichts-

Abbildung 2.8: Graph, auf dem der Greedy-Algorithmus für das Cliquen-Problem beliebig schlechte Lösungen liefert

funktion $c_1 = 1$, $c_2 = M > 2$, $c_3 = 2$. Das zugehörige Unabhängigkeitssystem hat die Basen $\{1,2\}$ und $\{3\}$. Bei Anwendung von (2.18) erhalten wir immer die Greedylösung $B_g = \{1,2\}$ mit Gewicht $1 + M$, während die Optimalbasis B_0 das Gewicht 2 hat. Der Quotient aus $c(B_g)$ und $c(B_0)$ geht also gegen ∞ falls $M \to \infty$. Denken Sie darüber nach, was diese (triviale) Beobachtung über die in Satz (2.14) angegebene Gütegarantie aussagt!

2.5 Ein primal-dualer Greedy-Algorithmus

Die Matroidtheorie ist aufgrund ihres technischen Apparates besonders gut dazu geeignet, eine gemeinsame Verallgemeinerung des primalen und des dualen Greedy-Algorithmus zur Berechnung minimaler aufspannender Bäume (siehe (5.9) und (5.11) im Skript zur ADM I) zu formulieren. Grundlegend hierfür ist der folgende

(2.20) Satz (und Definition). *Sei M ein Matroid auf einer Grundmenge E mit Basissystem $\mathcal{B}$. Setze $\mathcal{B}^* := \{E \setminus B \mid B \in \mathcal{B}\}$. Dann ist $\mathcal{B}^*$ das Basissystem eines Matroids M^* auf E. M^* wird das zu M* duale Matroid *genannt.* △

Offensichlich gilt für ein Basissystem

$$\mathcal{B}^{**} = \mathcal{B},$$

d. h. das zu M^* duale Matroid ist das Matroid M. Wir haben bereits ein Matroid und das dazu duale Matroid kennengelernt. Ist nämlich M das graphische Matroid auf einem Graphen G (siehe (2.8)(a)), so ist das cographische Matroid auf G (siehe (2.9)(a)) das zu M duale Matroid M^*.

Für Matrix-Matroide (siehe (2.8)(b)) kann man auf einfache Weise das duale Matroid konstruieren. Wir nehmen an, dass M durch die (m, n)-Matrix A (über irgendeinem Körper) definiert ist. O. B. d. A. können wir annehmen, dass A vollen Zeilenrang m hat. Mit Hilfe der Gauß-Elimination bringen wir A in **Standardform**, d. h. wir formen A so um, dass $A = (I, B)$ gilt, wobei I die (m, m)-Einheitsmatrix ist. (Möglichweise sind hierzu Zeilen- und Spaltenvertauschungen erforderlich.) Man kann nun zeigen, dass das zu M duale Matroid M^* durch die Matrix

$$(-B^T, I)$$

definiert ist. Hierbei ist I eine $(n-m, n-m)$-Einheitsmatrix. Das vor Satz (2.10) beschriebene aus der Geometrie stammende Fano-Matroid F_7 hat folglich als duales Matroid das Matroid, das durch die folgende Matrix

$$\begin{array}{c} \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

über den Körper $GF(2)$ gegeben ist. (Man beachte, dass $-1 = 1$ in $GF(2)$ gilt.) Man nennt es das *duale Fano-Matroid* und bezeichnet es mit F_7^*.

In der Matroidtheorie hat sich die folgende Sprechweise eingebürgert. Ist M ein Matroid und A eine Basis oder ein Zirkuit des dualen Matroids M^*, so nennt man A auch eine *Cobasis* oder ein *Cozirkuit* von M. Analog heißt der Rang von A bezüglich der Rangfunktion von M^* auch der *Corang* von A. Ein *Zyklus* ist die Vereinigung von paarweise disjunkten Zirkuits, analog ist ein *Cozyklus* die Vereinigung paarweise disjunkter Cozirkuits. Die nachfolgenden (einfachen) Beobachtungen sind für die anschließenden Betrachtungen wichtig.

(2.21) Satz.

(a) *Ein System $\mathcal{C}$ von Teilmengen von E ist genau dann die Zirkuitmenge eines Matroids M auf E, wenn $\mathcal{C}$ genau die minimalen, nichtleeren Teilmengen $C \subseteq E$ enthält, so dass gilt: $|C \cap C^*| \neq 1$ für alle Cozirkuits C^* von M.*

(b) *Sei B eine Basis (Cobasis) eines Matroids M, und sei $e \in E \setminus B$, dann enthält $B \cup \{e\}$ genau einen Zirkuit (Cozirkuit) C_e, der e enthält. C_e heißt der durch e und B erzeugte* fundamentale Zirkuit (Cozirkuit). △

(2.22) Satz. *Sei M ein Matroid auf E mit Gewichten c_e für alle $e \in E$. Sei B eine Basis von M. Dann sind die folgenden Aussagen äquivalent:*

(i) *B ist eine gewichtsminimale Basis.*

(ii) *Für jedes Element $e \in E \setminus B$ gilt $c_e \geq c_f$ für alle f aus dem von e und B erzeugten fundamentalen Zirkuit K_e.*

(iii) *Für jeden Cozirkuit C gilt* $\min\{c_e \mid e \in C\} = \min\{c_e \mid e \in B \cap C\}$

(iv) *$E \setminus B$ ist eine gewichtsmaximale Cobasis.*

(v) *Für jedes Element $e \in B$ gilt $c_e \leq c_f$ für alle f aus dem von e und $E \setminus B$ erzeugten fundamentalen Cozirkuit C_e.*

(vi) *Für jeden Zirkuit K gilt* $\max\{c_e \mid e \in K\} = \max\{c_e \mid e \in (E \setminus B) \cap K\}$. △

Beweis. (i) $\iff$ (iv). Trivial, nach Definition.

(i) $\implies$ (ii). Angenommen, es existieren $e \in E \setminus B$ und $f \in K_e$ mit $c_e < c_f$. Dann ist $(B \setminus \{f\}) \cup \{e\}$ eine Basis mit kleinerem Gewicht als B. Widerspruch!

(ii) $\implies$ (iii). Angenommen, es existieren ein Cozirkuit C und $e \in C \setminus B$ mit $c_e < \min\{c_f \mid f \in B \cap C\}$. Sei K_e der durch e mit B erzeugte fundamentale Zirkuit. Dann existiert nach (2.21)(a) ein Element $f \in (K_e \setminus \{e\}) \cap C$. Aber dann gilt für $e \in E \setminus B$ und $f \in B$: $c_e < c_f$. Widerspruch!

(iii) $\implies$ (i). Sei B' eine gewichtsminimale Basis, so dass $|B \cap B'|$ so groß wie möglich ist. Angenommen, es existiert ein Element $f \in B' \setminus B$. Sei C_f der von f mit $E \setminus B'$ erzeugte fundamentale Cozirkuit. Da B eine Basis ist, gilt $B \cap C_f \neq \emptyset$. Sei $g \in B \cap C_f$ mit $c_g = \min\{c_e \mid e \in B \cap C_f\}$. Nach (iii) gilt wegen $f \in C_f$: $c_f \geq c_g$. Nun aber ist $B'' := (B' \setminus \{f\}) \cup \{g\}$ eine Basis von M mit $c(B'') \leq c(B')$ und $|B \cap B''| > |B \cap B'|$. Widerspruch! Daraus folgt $B = B'$, und somit ist B gewichtsminimal.

(iv) $\implies$ (v) $\implies$ (vi) $\implies$ (iv) folgt aus Dualitätsgründen. □

(2.23) Algorithmus Primal-dualer Greedy-Algorithmus.
Eingabe: Grundmenge E mit Gewichten c_e für alle $e \in E$ und ein Matroid $M = (E, \mathcal{I})$.
Ausgabe: Eine gewichtsminimale Basis B von M und eine gewichtsmaximale Cobasis B^* von M.

(Terminologie: Wir sagen, dass eine Menge S durch eine Menge F *überdeckt* ist, falls $F \cap S \neq \emptyset$.)

1. Setze $B := B^* := \emptyset$.

2. Führe einen (beliebigen) der beiden folgenden Schritte 3 oder 4 aus:

3. Primaler Schritt
   3.1. Sind alle Cozirkuits von M durch B überdeckt, STOP.
   (Gib B und $B^* := E \setminus B$ aus.)
   3.2. Wähle einen Cozyklus $C \neq \emptyset$ von M, der nicht durch B überdeckt ist.
   3.3. Bestimme $f \in C$ mit $c_f = \min\{c_e \mid e \in C\}$.
   3.4. Setze $B := B \cup \{f\}$ und gehe zu 2.

4. Dualer Schritt
   4.1 Sind alle Zirkuits von M durch B^* überdeckt, STOP.
   (Gib B^* und $B := E \setminus B^*$ aus.)
   4.2 Wähle einen Zyklus $Z \neq \emptyset$ von M, der nicht durch B^* überdeckt ist.
   4.3 Bestimme $g \in Z$ mit $c_g = \max\{c_e \mid e \in Z\}$.
   4.4 Setze $B^* := B^* \cup \{g\}$ und gehe zu 2. △

Will man Algorithmus (2.23) implementieren, so muss entweder für Schritt 3.1 oder für Schritt 4.1 (oder für die beiden Schritte) ein Algorithmus vorhanden sein, der die verlangten Überprüfungen durchführt. Wir gehen hier (in der Theorie) davon aus, dass ein Orakel die Aufgabe für uns erledigt. Ob das Orakel effizient implementierbar ist, hängt natürlich vom Matroidtyp ab, den man behandeln will.

(2.24) Satz. *Der obige Algorithmus (2.23) funktioniert.* △

Beweis. Wir beweisen durch Induktion nach $|B| + |B^*|$, dass die jeweils während des Ablaufs des Verfahrens konstruierten Mengen B und B^* in einer gewichtsminimalen Basis bzw. gewichtsmaximalen Cobasis enthalten sind.

Für den Induktionsanfang $B = B^* = \emptyset$ ist die Behauptung trivial.

Wir nehmen an, dass der Satz gilt, wenn $|B| + |B^*| \leq k$, d. h. wenn wir Schritt 2 höchstens k-mal ausgeführt haben. Wir beginnen jetzt mit der $(k+1)$-ten Ausführung von Schritt 2 und entscheiden uns, Schritt 3 durchzuführen. Sei B die bisher konstruierte Menge und B' eine gewichtsminimale Basis mit $B \subseteq B'$. Wir wählen in 3.2 einen Cozyklus C und in 3.3 ein Element $f \in C$. Gilt $f \in B'$, so ist alles klar. Anderenfalls sei $g \in C \cap B'$ ein Element, das mit f auf einem Cozirkuit liegt. Dann muss wegen 3.3 gelten $c_g \geq c_f$. $B'' := (B' \setminus \{g\}) \cup \{f\}$ ist damit eine Basis mit $c(B'') \leq c(B')$ und $\{f\} \cup B \subseteq B''$. Also ist $B \cup \{f\}$ in einer gewichtsminimalen Basis enthalten.

Entscheiden wir uns in 2 für die Durchführung von Schritt 4, so folgt die Behauptung analog.

Stellen wir in 3.1 fest, dass B eine Basis ist, so ist nach Induktion $c(B)$ minimal und $E \setminus B$ eine maximale Cobasis. Analog schließen wir im Falle, dass B^* in 4.1 als Cobasis erkannt wird. □

Algorithmus (2.23) ist aufgrund der vielen Freiheitsgrade ein äußerst flexibel einsetzbares Instrument zur Konstruktion optimaler Basen und Cobasen. Durch geeignete Spezialisierung erhält man alle in Kapitel 5 des Skripts zur ADM I vorgestellten Algorithmen zur Bestimmung minimaler Bäume und einige weitere derartige Verfahren.

Literaturverzeichnis

F. Barahona und M. Grötschel. On the cycle polytope of a binary matroid. *Journal of Combinatorial Theory, Series B*, 40:40–62, 1986.

R. E. Bixby und W. H. Cunningham. Matroid optimization and algorithms. In R. L. Graham, M. Grötschel, L. Lovász, editor, *Handbook of Combinatorics, Volume I*, Kapitel 11, S. 551–609. North-Holland, Amsterdam, 1995.

J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1:127–136, 1971.

J. Edmonds. Matroid intersection. *Annals of Discrete Mathematics*, 4:39–49, 1979.

M. Grötschel und K. Truemper. Decomposition and Optimization over Cycles in Binary Matroids. *Journal of Combinatorial Theory, Series B*, 46(3):306–337, 1989.

D. Hausmann und B. Korte. The relative strength of oracles for independence systems. In J. Frehse, D. Pallaschke, und U. Trottenberg, editors, *Special topics of applied mathematics, functional analysis, numerical analysis, and optimization*, S. 195–211. North-Holland, Amsterdam, 1980.

D. Hausmann und B. Korte. Algorithmic versus axiomatic definitions of matroids. *Mathematical Programming Studies*, 14:98–111, 1981.

T. A. Jenkyns. The efficacy of the greedy algorithm. In F. Hoffman, L. Lesniak, und R. Mullin, editors, *Proceedings of the 7th Southeastern Conference on Combinatorics, Graph Theory and Computing*, Baton Rouge, 1976.

E. L. Lawler. Matroid intersection algorithms. *Mathematical Programming*, 9:31–56, 1975.

J. Lee. *A first course in combinatorial optimization.* University Press, Cambridge, 2004.

J. G. Oxley. *Matroid Theory.* Oxford University Press, Oxford, 1992.

A. Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency.* Springer-Verlag, Berlin, 2003.

K. Truemper. *Matroid Decomposition.* Academic Press, Boston, 1992.

D. J. A. Welsh. *Matroid Theory.* Academic Press, London, 1976.

D. J. A. Welsh. Matroids: Fundamental concepts. In R. L. Graham, M. Grötschel, L. Lovász, editor, *Handbook of Combinatorics, Volume I*, Kapitel 9, S. 481–526. North-Holland, Amsterdam, 1995.

3 Der LP/IP-Ansatz zur Lösung diskreter Optimierungsprobleme

3.1 Die allgemeine Methodik

Diskrete Optimierungsprobleme kommen in der Praxis auf zwei Arten vor. Zum einen werden sie graphentheoretisch (wie z. B. das Problem, einen kürzesten Weg zwischen zwei Knoten in einem Graphen zu finden) oder kombinatorisch (wie das Auffinden einer maximalen gewichteten Menge in einem Unabhängigkeitssystem) beschrieben, zum anderen ist ein lineares (oder nichtlineares) Optimierungsproblem gegeben, bei dem einige oder alle Variablen ganzzahlige Werte annehmen sollen. Bei der ersten Sorte spricht man häufig von *kombinatorischen Optimierungsproblemen*, bei der zweiten von *gemischt-ganzzahligen* oder *ganzzahligen Optimierungsproblemen*.

Für einige kombinatorische Optimierungsprobleme gibt es sehr schnelle und praxistaugliche Spezialalgorithmen. Die meisten anwendungsnahen Probleme sind jedoch $\mathcal{NP}$-schwer, und es ist nicht so richtig klar, wie man mit diesen algorithmisch umgehen soll. Darüber hinaus ist es häufig so, dass ein polynomial lösbares Problem schon bei einer harmlos aussehenden Zusatzbedingung schwer wird, und dann hilft das schnelle Spezialverfahren meistens nicht viel.

Ein üblicher und in der Praxis erfolgreicher Ansatz besteht darin, kombinatorische Optimierungsprobleme als ganzzahlige Optimierungsprobleme zu modellieren und zur Lösung die „Maschinerie“ der linearen Optimierung einzusetzen. Dies geschieht dadurch, dass man aus kombinatorischen Problemen geometrische macht, und zwar wird jeder zulässigen Lösung des gegebenen kombinatorischen Problems ein Vektor (genannt *Inzidenzvektor*) in einem „passenden“ Vektorraum zugeordnet. Hier gibt es verschiedene Vorgehensweisen (Modellierungstechniken). Der übliche „Trick“ besteht darin, Inzidenzvektoren so zu definieren, dass ihre konvexe Hülle ein Polytop bildet, dessen Ecken genau die Inzidenzvektoren der zulässigen Lösungen des kombinatorischen Optimierungsproblems sind. Ist ein solches Polytop gefunden, verbleibt noch die Aufgabe, eine Beschreibung des Polytops durch lineare Gleichungen und Ungleichungen zu finden. Dies ist im Prinzip möglich (wie wir aus ADM I wissen), aber in den meisten Fällen schwierig. In diesem Kapitel wollen wir einige Beispiele angeben, bei denen eine vollständige oder partielle und praktisch nutzbringende Beschreibung eines solchen Polytops gelungen ist.

Es ist verblüffend, dass selbst für ganz einfache kombinatorische Optimierungsprobleme die zugehörigen Polytope sehr komplex sein können. So kommt es durchaus häufig vor, dass die Anzahl der Facetten des Polytops exponentiell in der Anzahl der Grunddaten (z. B. Knoten des Graphen, Elemente des Unabhängigkeitssystems) ist. Trotzdem kann man – wie wir später sehen werden – lineare Programme über solchen Ungleichungssys-

temen gelegentlich in polynomialer Zeit lösen. Aber selbst dann, wenn eine vollständige Darstellung eines solchen Polytops nicht geglückt ist, sind partielle Beschreibungen häufig in Schnittebenenverfahren sehr nützlich. Darauf werden wir später genauer eingehen.

Jetzt fangen wir erst einmal mit „schönen Beispielen“ an und führen so in das reizvolle Forschungsgebiet „polyedrische Kombinatorik“ ein, welches die Grundlage für viele praktische Erfolge der kombinatorischen Optimierung bildet.

3.2 Unabhängigkeitssysteme und Polyeder, Matroid-Polytope

Wir wenden uns nun den vorher angedeuteten LP/IP-Aspekten zu und werden untersuchen, wie man Optimierungsprobleme über Unabhängigkeitssystemen als ganzzahlige Programme formulieren kann. Ein besonderes Highlight wird die vollständige Charakterisierung von Matroid-Polytopen sein.

Wir beginnen mit einem beliebigen Unabhängigkeitssystem $(E, \mathcal{I})$ auf einer endlichen Grundmenge E. Jedem Element $e \in E$ ordnen wir wie üblich eine Variable x_e zu. Nimmt x_e den Wert 1 an, so interpretieren wir dies als Wahl des Elements $e \in E$, falls $x_e = 0$, so entscheiden wir uns gegen die Wahl von e. Für jede Teilmenge $F \subseteq E$ führen wir wie üblich den Inzidenzvektor $\chi^F = (\chi^F_e)_{e \in E} \in \mathbb{K}^E$ ein, dessen Komponenten wie folgt definiert sind: $\chi^F_e = 1$, falls $e \in F$ und $\chi^F_e = 0$, falls $e \notin F$.

Dem Unabhängigkeitssystem $\mathcal{I}$ ordnen wir ein Polytop $\mathrm{IND}(\mathcal{I}) \subseteq \mathbb{K}^E$ wie folgt zu:

$$\mathrm{IND}(\mathcal{I}) := \mathrm{conv}\{\chi^I \in \mathbb{K}^E \mid I \in \mathcal{I}\}. \tag{3.1}$$

Ist $(E, \mathcal{I})$ ein Matroid, so nennt man $\mathrm{IND}(\mathcal{I})$ *Matroid-Polytop*.

Offensichtlich kann man bei gegebener linearer Zielfunktion $c \in \mathbb{K}^E$ das kombinatorische Optimierungsproblem $\max\{c(I) \mid I \in \mathcal{I}\}$ als lineares Programm $\max\{c^T x \mid x \in \mathrm{IND}(\mathcal{I})\}$ lösen. (Jedes LP über dem Polytop $\mathrm{IND}(\mathcal{I})$ hat eine optimale Ecklösung, und diese ist nach Definition der Inzidenzvektor einer unabhängigen Menge $I \in \mathcal{I}$.) Um LP-Methoden anwenden zu können, müssen wir jedoch lineare Ungleichungen finden, die $\mathrm{IND}(\mathcal{I})$ vollständig (oder zumindest partiell) beschreiben.

Zunächst entledigen wir uns einer Trivialität. In einem Unabhängigkeitssystem kann es Elemente $e \in E$ geben, so dass $\{e\}$ keine unabhängige Menge ist. Solche Elemente heißen *Schlingen*, *Schleifen* oder *Loops*. Sie können niemals in einer optimalen unabhängigen Menge enthalten sein. Wir nehmen daher im Folgenden o. B. d. A. an, dass $(E, \mathcal{I})$ keine Schlingen enthält. (In der Literatur nennt man solche Unabhängigkeitssysteme *normal*.)

Die folgende einfache Überlegung führt uns auf die Spur eines wichtigen Ungleichungssystems für $\mathrm{IND}(\mathcal{I})$. Ist $F \subseteq E$, so ist der Rang $r(F)$ von F die größte Kardinalität einer in F enthaltenen unabhängigen Menge. Falls $F \in \mathcal{I}$, so gilt natürlich $r(F) = |F|$. Für jede unabhängige Menge $I \in \mathcal{I}$ ist $F \cap I \in \mathcal{I}$, und daher gilt $|F \cap I| \leq r(F)$. Daraus folgt, dass jeder Inzidenzvektor χ^I einer unabhängigen Menge I die Ungleichung (genannt *Rang-Ungleichung*)

$$\sum_{e \in F} x_e = x(F) \leq r(F) \tag{3.2}$$

erfüllt. Dabei ist $x(\{e\}) \leq r(\{e\})$ nichts anderes als die triviale Ungleichung $x_e \leq 1$, für alle $e \in E$.

(3.3) Satz. *Seien $(E, \mathcal{I})$ ein Unabhängigkeitssystem und $c \in \mathbb{K}^E$. Dann gilt:*

(a) $\text{IND}(\mathcal{I}) \subseteq P(\mathcal{I}) := \{x \in \mathbb{K}^E \mid x(F) \leq r(F)\ \forall F \subseteq E,\ x_e \geq 0\ \forall e \in E\}$.

(b) $$\max\{c(I) \mid I \in \mathcal{I}\} \;=\; \begin{array}{lll} \max & c^T x & \\ \text{s.t.} & x(F) \leq r(F) & \forall F \subseteq E \\ & x_e \geq 0 & \forall e \in E \\ & x_e \in \{0,1\} & \forall e \in E. \end{array}$$

△

Beweis. Wie bereits erläutert, erfüllt jeder Vektor χ^I, $I \in \mathcal{I}$, das obige Ungleichungssystem, somit ist jede Ecke von $\text{IND}(\mathcal{I})$ in $P(\mathcal{I})$ enthalten, und daher gilt $\text{IND}(\mathcal{I}) \subseteq P(\mathcal{I})$; (a) ist damit bewiesen.

Um (b) zu zeigen, müssen wir beweisen, dass die Vektoren χ^I, $I \in \mathcal{I}$ die einzigen ganzzahligen Vektoren in $P(\mathcal{I})$ sind. Sei also $y \in P(\mathcal{I})$ ganzzahlig, dann muss y ein 0/1-Vektor und somit Inzidenzvektor einer Teilmenge $F \subseteq E$ sein. Ist $F \notin \mathcal{I}$, so gilt $r(F) < |F| = \sum_{e \in F} y_e$ und somit erfüllt y nicht die Ungleichung $x(F) \leq r(F)$. Also muss F unabhängig sein, und der Beweis ist erledigt. □

Satz (3.3)(b) gibt uns somit die Möglichkeit, beliebige lineare Optimierungsprobleme über Unabhängigkeitssystemen als ganzzahlige Programme zu lösen. Natürlich ist zu bemerken, dass das System aus Satz (3.3) insgesamt $2^m + m$ Ungleichungen enthält ($|E| = m$). Diese „Schwierigkeit“ werden wir später behandeln.

Ist das System vollständig, d. h., gilt $\text{IND}(\mathcal{I}) = P(\mathcal{I})$? Wir betrachten dazu das Stabile-Mengen-Problem auf dem Graphen $G = (V, E)$ aus Abbildung 3.1. Das Unabhängigkeits-

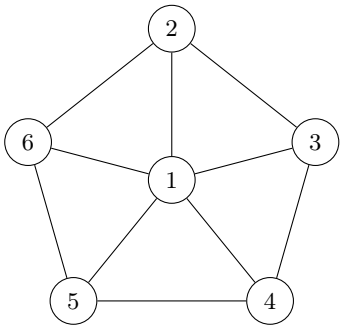


Abbildung 3.1: Graph mit 6 Knoten

system ist auf der Knotenmenge $V = \{1, \ldots, 6\}$ definiert. Die stabilen Knotenmengen bilden das System unabhängiger Mengen $\mathcal{I} \subseteq 2^V$. Offensichtlich sind nur die leere Menge, die Mengen $\{v\}$, $v = 1, \ldots, 6$ und $\{2,4\}, \{2,5\}, \{3,5\}, \{3,6\}, \{4,6\}$ stabile Knotenmengen. Betrachten wir die Zielfunktion $c \in \mathbb{K}^6$ mit $c_1 = 2$, $c_i = 1$, $i = 2, \ldots, 6$, so hat das

Optimum von $c^T x$ über $\mathrm{IND}(\mathcal{I})$ offenbar den Wert 2. Der Vektor $x^T = (x_1, \ldots, x_6)$ mit $x_1 = \cdots = x_6 = \frac{1}{3}$ hat den Zielfunktionswert $\frac{7}{3}$ und erfüllt offenbar alle Rangungleichungen, ist also in $P(\mathcal{I})$ enthalten. Daraus folgt $P(\mathcal{I}) \neq \mathrm{IND}(\mathcal{I})$ für dieses Unabhängigkeitssystem.

Wir beweisen nun, dass die Rangungleichungen zur Beschreibung von Matroid-Polytopen ausreichen.

(3.4) Satz. *Sei $(E, \mathcal{I})$ ein Matroid, dann gilt:*

$$P(\mathcal{I}) = \mathrm{IND}(\mathcal{I}). \qquad \triangle$$

Beweis. Nach Annahme ist $(E, \mathcal{I})$ normal. Damit enthält $\mathrm{IND}(\mathcal{I})$ neben dem Nullvektor $\chi^{\emptyset}$ auch alle Einheitsvektoren $\chi^{\{e\}}$, $e \in E$. Mithin ist $\mathrm{IND}(\mathcal{I})$ volldimensional ($\dim \mathrm{IND}(\mathcal{I}) = |E|$) und alle Facetten definierenden Ungleichungen von $\mathrm{IND}(\mathcal{I})$ sind bis auf positive Vielfache eindeutig bestimmt, siehe Korollar (1.40).

Sei nun $a^T x \leq b$ eine gültige Ungleichung, die eine Facette $H := \mathrm{IND}(\mathcal{I}) \cap \{x \in \mathbb{K}^E \mid a^T x = b\}$ von $\mathrm{IND}(\mathcal{I})$ definiert. Wir werden zeigen, dass dann $a^T x \leq b$ ein positives Vielfaches von einer der Ungleichungen $-x_e \leq 0$, $e \in E$, oder $x(F) \leq r(F)$, $F \subseteq E$ ist. Das bedeutet, dass das $P(\mathcal{I})$ definierende System alle Ungleichungen enthält, die Facetten von $\mathrm{IND}(\mathcal{I})$ definieren, und somit gilt $P(\mathcal{I}) = \mathrm{IND}(\mathcal{I})$.

Wir nehmen zunächst an, dass ein Koeffizient von a negativ ist, sagen wir $a_e < 0$, $e \in E$. Gibt es eine unabhängige Menge $I \in \mathcal{I}$ mit $e \in I$ und $a^T \chi^I = b$, dann gilt für die ebenfalls unabhängige Menge $J := I \setminus \{e\}$: $a^T \chi^J = a^T \chi^I - a_e = b - a_e > b$. Dies widerspricht der Gültigkeit von $a^T x \leq b$. Folglich gilt $x_e = 0$ für alle Punkte der Facette H und damit, dass

$$H \subseteq \mathrm{IND}(\mathcal{I}) \cap \{x \in \mathbb{K}^E \mid x_e = 0\} =: G.$$

Da H eine maximale Seitenfläche von $\mathrm{IND}(\mathcal{I})$ ist, folgt $H = G$, und aus (1.40) folgt somit, dass $a^T x \leq b$ ein positives Vielfaches von $-x_e \leq 0$ ist.

Nehmen wir nun an, dass $a_e \geq 0$ für alle $e \in E$ gilt. Setze $F := \{e \in E \mid a_e > 0\}$. Wir behaupten nun, dass $a^T x \leq b$ ein positives Vielfaches von $x(F) \leq r(F)$ ist. Um das zu beweisen, zeigen wir, dass, falls $I \in \mathcal{I}$ und $a^T \chi^I = b$ gilt, dann gilt auch $\chi^I(F) = r(F)$, d. h. dass $I \cap F$ eine Basis von F ist. Hieraus folgt wie oben die Behauptung.

Seien also $I \in \mathcal{I}$ und $a^T \chi^I = b$. Angenommen es gibt ein Element $e \in F \setminus I$, sodass $J := (I \cap F) \cup \{e\} \in \mathcal{I}$, dann gilt $a^T \chi^J = a^T \chi^I + a_e > a^T \chi^I = b$. Dies widerspricht der Gültigkeit von $a^T x \leq b$. Da das Hinzufügen eines jeden Elements $e \in F \setminus I$ zu $F \cap I$ eine abhängige Menge liefert, ist $I \cap F$ eine Basis von F und somit erfüllt I die Rangungleichung $x(F) \leq r(F)$ mit Gleichheit. □

Man kann sogar die Facetten von Matroid-Polytopen vollständig charakterisieren. Wir benötigen hierzu zwei neue Begriffe. Eine Teilmenge $F \subseteq E$ heißt *abgeschlossen*, wenn

$$r(F \cup \{e\}) > r(F) \qquad \forall e \in E \setminus F$$

gilt. F heißt *separabel*, wenn es zwei nichtleere Teilmengen $F_1, F_2 \subseteq F$ mit $F_1 \cup F_2 = F$ und $F_1 \cap F_2 = \emptyset$ gibt, sodass

$$r(F) = r(F_1) + r(F_2),$$

anderenfalls heißt F *inseparabel*.

(3.5) Satz. *Sei $(E, \mathcal{I})$ ein normales Matroid.*

(a) *Eine Rangungleichung $x(F) \leq r(F)$, $F \subseteq E$ definiert eine Facette von* $\mathrm{IND}(\mathcal{I})$ *genau dann, wenn F abgeschlossen und inseparabel ist.*

(b) *Das folgende System von Ungleichungen ist eine vollständige und irredundante Beschreibung des Matroid-Polytops* $\mathrm{IND}(\mathcal{I})$*:*

$$\begin{aligned} x(F) &\leq r(F) && \forall F \subseteq E,\ F \text{ abgeschlossen und inseparabel} \\ x_e &\geq 0 && \forall e \in E. \end{aligned}$$

△

Der Beweis von Satz (3.5) ist umfangreich und kann aus Zeitgründen hier nicht vorgeführt werden.

(3.6) Bemerkung.

(a) Sei $G = (V, E)$ ein schlingenfreier Graph. Eine Teilmenge $F \subseteq E$ ist abgeschlossen und inseparabel im graphischen Matroid auf E genau dann, wenn entweder $F = \{e\}$, $e \in E$ gilt oder wenn F die Kantenmenge eines knoteninduzierten Untergraphen $(W, E(W))$ mit $|W| \geq 3$ ist, der 2-fach knotenzusammenhängend ist.

(b) Sei $(E, \mathcal{I})$ das Partitionsmatroid auf E, das definiert ist durch $E_1, \ldots, E_k \subseteq E$ und $b_1, \ldots, b_k$ mit $1 \leq b_i < |E_i|$ für alle $i = 1, \ldots, k$, siehe (2.9)(c). $F \subseteq E$ ist genau dann abgeschlossen und inseparabel in $\mathcal{I}$, wenn $F = E_i$ für ein $i \in \{1, \ldots, k\}$ oder $F = \{e\}$ für ein $e \in E_i$ mit $b_i \geq 2$, $i \in \{1, \ldots, k\}$ gilt. △

Beweis. Übungsaufgabe. □

Eine Anwendung von Satz (3.5) unter der Berücksichtigung der Erkenntnisse in (3.6) ist der folgende Satz.

(3.7) Satz.

(a) *Das Matroid-Polytop eines graphischen Matroids auf einem schlingenfreien Graphen $G = (V, E)$ (das Polytop der Wälder) ist vollständig und irredundant charakterisiert durch das folgende Ungleichungssystem:*

$$\begin{aligned} x(E(W)) &\leq |W| - 1 && \forall\, W \subseteq V, |W| \geq 3, \\ & && \quad (W, E(W)) \text{ 2-fach knotenzusammenhängend} \\ 0 \leq x_e &\leq 1 && \forall\, e \in E. \end{aligned}$$

(b) *Das Matroid-Polytop eines Partitionsmatroids auf einer endlichen Menge E, gegeben durch $E_1, \dots, E_k$ und $b_1, \dots, b_k$ mit $1 \le b_i < |E_i|$ für alle $i = 1, \dots, k$, ist vollständig und irredundant charakterisiert durch das folgende Ungleichungssystem:*

$$\begin{aligned} x(E_i) &\le b_i && i = 1, \dots, k \\ x_e &\le 1 && \forall\, e \in E_i \text{ mit } b_i \ge 2 \\ x_e &\ge 0 && \forall\, e \in E_i. \end{aligned}$$ △

Satz (3.7) zeigt eine mögliche dramatische Reduktion der Anzahl der zur Beschreibung eines Matroid-Polytops benötigten Ungleichungen. In Satz (3.4) wird bewiesen, dass die in Satz (3.3) angegebenen Ungleichungen zur Beschreibung ausreichen. Die Anzahl dieser Ungleichungen ist $2^m + m$, mit $m = |E|$. Satz (3.7) zeigt, dass diese Zahl bei vollständigen Graphen auf ungefähr 2^n, $n = |V|$ reduziert werden kann. Sie ist jedoch immer noch exponentiell in der Kodierungslänge des Graphen. Bei „sehr dünnen“ Graphen ist die Anzahl der Ungleichungen erheblich kleiner. Bei einem Partitionsmatroid werden statt $2^m + m$ höchstens $k + 2m$ Ungleichungen benötigt.

Es gibt noch eine erstaunliche Verschärfung von Satz (3.5), die wir ebenfalls ohne Beweis angeben.

(3.8) Satz. *Sei E eine endliche Menge. $(E, \mathcal{I}_1)$ und $(E, \mathcal{I}_2)$ seien zwei Matroide auf E. Das Polytop des Durchschnitts dieser beiden Matroide*

$$P(\mathcal{I}_1 \cap \mathcal{I}_2) = \operatorname{conv}\{\chi^I \in \mathbb{K}^E \mid I \in \mathcal{I}_1 \cap \mathcal{I}_2\}$$

wird durch das folgende System von Ungleichungen vollständig beschrieben:

$$\begin{aligned} x(F) &\le \min\{r_1(F), r_2(F)\} && \forall F \subseteq E \\ x_e &\ge 0 && \forall e \in E, \end{aligned}$$

wobei $r_i(F)$ die Rangfunktion von $(E, \mathcal{I}_i)$, $i = 1, 2$ bezeichnet. △

An dieser Stelle sollen noch einmal Satz (2.14) und der Beweis aufgenommen und mit den „LP-Erkenntnissen“ dieses Abschnitts verbunden werden. Für ein beliebiges Unabhängigkeitssystem $(E, \mathcal{I})$ und eine beliebige Zielfunktion $c \in \mathbb{K}^E$ bezeichnet I_0 eine optimale Lösung des so definierten Optimierungsproblems $\max\{c(I) \mid I \in \mathcal{I}\}$, I_g bezeichnet die Greedy-Lösung, $E_1, \dots, E_n$ seien die im Beweis von (2.14) definierten Mengen $E_i = \{1, \dots, i\}$, $i = 1, \dots, n$ (unter der Annahme $c_1 \ge c_2 \ge \cdots \ge c_n \ge c_{n+1} := 0$) und q bezeichnet den Rangquotienten. Mit $y^* \in \mathbb{K}^{2^E}$, definiert durch

$$y^*_F := \begin{cases} c_i - c_{i+1} & \text{für } F = E_i \\ 0 & \text{sonst} \end{cases}$$

gilt dann die folgende Abschätzung:

$$
\begin{aligned}
&\max \sum_{e\in E} c_e x_e,\ \sum_{e\in F} x_e \le r(F)\ \forall F\subseteq E, x_e \ge 0\ \forall e\in E\\
\ge\ &\max \sum_{e\in E} c_e x_e,\ \sum_{e\in F} x_e \le r(F)\ \forall F\subseteq E, x_e \ge 0\ \forall e\in E,\ x \text{ ganzzahlig}\\
=\ & c(I_0)\ \ge\ c(I_g)\\
=\ & \sum_{i=1}^{n}(c_i - c_{i+1})|I_g\cap E_i|\ \ge\ \sum_{i=1}^{n}(c_i-c_{i+1})\, r_u(E_i)\ =\ \sum_{i=1}^{n} y^*_{E_i} r_u(E_i)\\
\ge\ &\min \sum_{F\subseteq E} y_F\, r_u(F),\ \sum_{F\ni e} y_F \ge c_e\ \forall e\in E, y_F\ge 0\ \forall F\subseteq E\\
\ge\ & q\ \min \sum_{F\subseteq E} y_F\, r(F),\ \sum_{F\ni e} y_F \ge c_e\ \forall e\in E, y_F\ge 0\ \forall F\subseteq E\\
=\ & q\ \max \sum_{e\in E} c_e x_e,\ \sum_{e\in F} x_e \le r(F)\ \forall F\subseteq E, x_e \ge 0\ \forall e\in E\\
\ge\ & q\ \max \sum_{e\in E} c_e x_e,\ \sum_{e\in F} x_e \le r(F)\ \forall F\subseteq E, x_e \ge 0\ \forall e\in E,\ x \text{ ganzzahlig}\\
=\ & q\, c(I_0)
\end{aligned}
$$

Satz (2.14) beweist

$$c(I_g)\ \ge\ q\, c(I_0).$$

Die obige Abschätzung liefert die Verbesserung

$$c(I_g)\ \ge\ q\ \max\{c^T x \mid x\in P(\mathcal{I})\}, \tag{3.9}$$

d. h. der Wert der Greedy-Lösung ist garantiert so groß wie das q-fache der LP-Relaxierung des kombinatorischen Optimierungsproblems.

Anwendungen dieser sehr allgemeinen Theorie auf einige konkrete kombinatorische Optimierungsprobleme mit praktischer Relevanz werden in den Übungen besprochen.

3.3 Matching-Polyeder

Wir erinnern daran, dass eine Kantenmenge M in einem Graphen $G=(V,E)$ als *Matching* bezeichnet wird, wenn jeder Knoten $v\in V$ in höchstens einer Kante $e\in M$ enthalten ist. Ein Matching heißt *perfekt*, wenn jeder Knoten in (genau) einer Kante enthalten ist. Sind Kantengewichte c_e für alle $e\in E$ gegeben, so interessiert man sich typischerweise für ein Matching M mit maximalem Gewicht $\sum_{e\in M} c_e$ oder für ein perfektes Matching minimalen Gewichts.

Wie üblich ordnen wir jedem Matching M einen Inzidenzvektor $\chi^M=(\chi^M_e)_{e\in E}$ mit $\chi^M_e=1$, falls $e\in M$ und $\chi^M_e=0$, falls $e\notin M$ zu und können mit diesen Vektoren zwei

Polyeder definieren:

$$\text{MATCH}(G) := \text{conv}\{\chi^M \in \mathbb{K}^E \mid M \subseteq E \text{ Matching}\}, \tag{3.10}$$

$$\text{PMATCH}(G) := \text{conv}\{\chi^M \in \mathbb{K}^E \mid M \subseteq E \text{ perfektes Matching}\}, \tag{3.11}$$

deren Ecken genau die Inzidenzvektoren der (perfekten) Matchings sind. MATCH(G) wird das *Matching-Polytop* von G genannt, und PMATCH(G) heißt *perfektes Matching-Polytop*. Ein Graph mit ungerader Knotenzahl enthält natürlich kein perfektes Matching, dieses Problem ist daher nur für Graphen mit gerader Knotenzahl von Interesse, aber auch in einem solchen Fall kann PMATCH(G) natürlich leer sein.

Bei der Suche nach Gleichungen und Ungleichungen, die von allen Inzidenzvektoren von Matchings erfüllt werden, wird man schnell fündig. Betrachten wir z. B.

$$\begin{aligned} x(\delta(v)) &\leq 1 \qquad \forall v \in V \\ x_e &\geq 0 \qquad \forall e \in E \end{aligned} \tag{3.12}$$

und

$$\begin{aligned} x(\delta(v)) &= 1 \qquad \forall v \in V \\ x_e &\geq 0 \qquad \forall e \in E \end{aligned} \tag{3.13}$$

(wobei $\delta(v) = \{e \in E \mid v \text{ ist Endknoten von } e\}$), dann ist offensichtlich, dass jeder Inzidenzvektor eines Matchings das System (3.12) und jeder Inzidenzvektor eines perfekten Matchings das System (3.13) erfüllt. (Ungleichungen des Typs $x(\delta(v)) \leq k$ kommen in der kombinatorischen Optimierung häufig vor. Sie werden *Gradungleichungen* genannt.) Ferner sieht man mühelos ein, dass die Inzidenzvektoren von (perfekten) Matchings genau die 0/1-Vektoren sind, die (3.13) bzw. (3.12) erfüllen. Fügen wir also zu (3.12) und (3.13) Zielfunktionen $\max c^T x$ bzw. $\min c^T x$ und Ganzzahligkeitsbedingungen hinzu, so haben wir eine Formulierung des (perfekten) Matching-Problems als ganzzahliges Programm gefunden.

Jetzt erinneren wir uns daran, dass die Nebenbedingungsmatrix des Systems (3.12) bzw. (3.13) die $|V| \times |E|$-Inzidenzmatrix des gegebenen Graphen $G = (V, E)$ ist und dass diese, siehe Beispiel (13.11) aus ADM I, eine total unimodulare Matrix ist, falls G bipartit ist. Daraus folgt

(3.14) Satz. *Sei $G = (V, E)$ ein bipartiter Graph, dann gilt*

(a) $\text{MATCH}(G) = \{x \in \mathbb{K}^E \mid x(\delta(v)) \leq 1 \ \forall v \in V, x \geq 0\}$,

(b) $\text{PMATCH}(G) = \{x \in \mathbb{K}^E \mid x(\delta(v)) = 1 \ \forall v \in V, x \geq 0\}$. △

Hierbei folgt (3.14)(a) direkt aus dem Satz von Hoffman und Kruskal (13.9) und (b) aus Satz (13.8) aus ADM I. Aussage (b) folgt natürlich auch direkt aus (3.14)(a), denn PMATCH(G) ist offensichtlich eine Seitenfläche von MATCH(G), und wenn ein Polyeder ganzzahlig ist, dann gilt das auch für jede Seitenfläche.

Satz (3.14) gilt nicht für Graphen, die nicht bipartit sind. Wir erinnern uns daran, dass ein Graph bipartit ist, wenn er keinen Kreis ungerader Länge enthält. Sei nun C ein ungerader Kreis, dann definieren wir den Vektor $y = (y_e)_{e \in E}$ wie folgt:

$$y_e := \frac{1}{2} \quad \forall e \in C, \qquad y_e := 0 \quad \forall e \in E \setminus C.$$

Der Vektor y erfüllt das Ungleichungssystem (3.12), ist aber nicht in MATCH(G) enthalten. Das kann man folgendermaßen einsehen. Wir definieren die Zielfunktion $c = (c_e)_{e \in E}$ durch

$$c_e := 1 \quad \forall e \in C, \qquad c_e := 0 \quad \forall e \in E \setminus C.$$

Es ist unmittelbar klar, dass das Maximum dieser Zielfunktion über MATCH(G) den Wert $(|C| - 1)/2$ hat, während $c^T y = |C|/2$ gilt. Also liegt y nicht in MATCH(G).

Wenn wir MATCH(G) vollständig durch lineare Ungleichungen beschreiben wollen, benötigen wir zusätzliche Ungleichungen. Und jetzt nutzen wir aus, dass die Matchings eines Graphen ein Unabhängigkeitssystem bilden und dass wir in Abschnitt 3.2 die Rangungleichungen kennengelernt haben. Sei $W \subseteq V$ eine Knotenmenge ungerader Kardinalität. Ist $E(W)$ die Menge aller Kanten mit beiden Endknoten in W, so gilt offensichtlich $M \cap E(W) \leq (|W| - 1)/2$ für alle Matchings M. Mit anderen Worten, der Rang von $E(W)$ bezüglich des Unabhängigkeitssystems aller Matchings ist $(|W| - 1)/2$. Daraus ergibt sich, dass das folgende System von Rangungleichungen

$$x(E(W)) \leq (|W| - 1)/2 \qquad \forall W \subseteq V, |W| \text{ ungerade} \tag{3.15}$$

von allen Punkten in MATCH(G) erfüllt wird. In der Literatur heißen die Ungleichungen (3.15) *Matching-Ungleichungen.*

Bezüglich des perfekten Matching-Polytops kann man die Ungleichungen (3.15) in eine andere Form bringen. Für $W \subseteq V$, $|W|$ ungerade, betrachten wir die mit -2 skalierte Matching-Ungleichung

$$-2x(E(W)) \geq -|W| + 1$$

und addieren zu dieser die $|W|$ Gleichungen $x(\delta(v)) = 1$, $v \in W$. Das Ergebnis ist die Ungleichung

$$x(\delta(W)) \geq 1. \tag{3.16}$$

Ungleichungen des Typs (3.16) heißen in der Literatur *ungerade Schnitt-Ungleichungen* (odd cut constraints). Bezüglich PMATCH(G) sind die beiden Ungleichungen $x(\delta(W)) \geq 1$ und $x(E(W)) \leq (|W| - 1)/2$ äquivalent. Das heißt, die beiden verschiedenen Ungleichungen definieren dieselbe Seitenfläche von PMATCH(G). Das gilt aber nicht für MATCH(G), denn $x(\delta(W)) \geq 1$ ist für MATCH(G) überhaupt nicht gültig. Ein bedeutendes Resultat der polyedrischen Kombinatorik ist der folgende Satz, der in Edmonds (1965) (mit einem anderen Beweis) bewiesen wurde.

(3.17) Satz (Das perfekte Matching-Polytop-Theorem von Edmonds). *Das perfekte Matching-Polytop* PMATCH(G) *eines Graphen* $G = (V, E)$ *ist durch das folgende System von Gleichungen und Ungleichungen vollständig beschrieben.*

(i) $x(\delta(v)) = 1 \quad \forall v \in V$

(ii) $x(\delta(W)) \geq 1 \quad \forall W \subseteq V$, $|W|$ *ungerade*

(iii) $x_e \geq 0 \quad \forall e \in E.$ △

Zunächst eine Bemerkung. Die Ungleichungen $x(\delta(W)) \geq 1$ in (ii) mit $|W| = 1$ können wir natürlich weglassen, da sie von (i) impliziert werden. Ist $|V|$ ungerade, so enthält (ii) die Ungleichung $x(\delta(V)) \geq 1$, $\delta(V)$ ist aber die leere Menge, und so lautet die Ungleichung $0^T x \geq 1$, woraus folgt, dass PMATCH(G) leer ist, falls $|V|$ ungerade ist. Wir können daher im Weiteren annehmen, dass $|V|$ gerade ist. Wegen $\delta(W) = \delta(V \setminus W)$ enthält (ii) jede Ungleichung zweimal. Wir können das z. B. dadurch verhindern, dass wir in (ii) zusätzlich fordern, dass W immer einen speziellen Knoten enhält. Für den nachfolgenden Beweis ist das jedoch unerheblich.

Beweis (von (3.17)). Sei $P(G)$ das Polytop, das durch (i), (ii), (iii) definiert ist. Wir haben uns bereits überlegt, dass jeder Inzidenzvektor eines perfekten Matchings die Ungleichungen (i), (ii), (iii) erfüllt, also gilt PMATCH(G) $\subseteq P(G)$.

Um $P(G) \subseteq$ PMATCH(G) zu zeigen, nehmen wir an, dass diese Behauptung falsch ist. Wenn das so ist, gibt es einen Graphen $G = (V, E)$ mit $P(G) \not\subseteq$ PMATCH(G) und mit der Eigenschaft, dass $|V|+|E|$ so klein wie möglich ist (ein kleinstes Gegenbeispiel). $P(G)$ muss dann eine Ecke y haben, die nicht in PMATCH(G) ist. Unsere Annahme impliziert, dass $0 < y_e < 1$ $\forall e \in E$ gilt. Das sieht man ohne Mühe durch zwei Widerspruchsbeweise wie folgt ein.

Gäbe es eine Kante $e \in E$ mit $y_e = 0$, so definieren wir den Graphen $G' = (V, E')$, $E' := E \setminus \{e\}$ und den Vektor $y' := (y_e)_{e \in E'}$. Satz (8.8) aus ADM I impliziert unmittelbar, dass y' eine Ecke von $P(G')$ ist. Somit hätten wir ein kleineres Gegenbeispiel gefunden.

Gäbe es eine Kante $e \in E$, $e = uv$ mit $y_e = 1$, so betrachten wir den Graphen $G' = (V, E')$ mit $E' := E \setminus (\delta(u) \cup \delta(v))$ und $y' := (y_e)_{e \in E'}$. Analysiert man die Ungleichungen, die weiterhin mit Gleichheit erfüllt sind im Detail, so liefert auch in diesem Fall Satz (8.8) aus ADM I, dass y' eine Ecke von $P(G')$ ist und damit ein kleineres Gegenbeispiel liefert.

Wir können noch weitere einfache Eigenschaften des minimalen Gegenbeispiels ableiten. G enthält keine isolierten Knoten, denn diese könnten wir aus G entfernen und hätten ein kleineres Gegenbeispiel. G hat keinen Knoten mit Grad 1, denn die Kante $e \in E$, die diesen Endknoten enthält, hätte wegen Gleichung (i) immer den Wert $y_e = 1$. Und schließlich haben nicht alle Knoten von G den Grad 2, denn dann wäre G die disjunkte Vereinigung von Kreisen, und für derartige Graphen ist der Satz offensichtlich richtig. Daraus folgt insbesondere $|E| > |V|$.

Da y eine Ecke von $P(G)$ ist, gibt es nach Satz (8.8) aus ADM I $|E|$ linear unabhängige Nebenbedingungen (i), (ii), die von y mit Gleichheit erfüllt werden. Insbesondere gibt es wegen $|E| > |V|$ mindestens eine ungerade Knotenmenge $W \subseteq V$ mit der Eigenschaft, dass $|W| \geq 3$, $|V \setminus W| \geq 3$ und $y(\delta(W)) = 1$ gilt.

Wir erinnern nun an die im Abschnitt 2.1 von ADM I definierte *Kontraktion* einer Knotenmenge S. Alle Knoten aus S werden aus dem Graphen entfernt und durch einen einzigen neuen Knoten s ersetzt. Alle Kanten aus $E(S)$ werden entfernt, alle Kanten aus

$E(V \setminus S)$ bleiben erhalten, und alle Kanten aus $\delta(S)$ bleiben erhalten, jedoch wird in jeder Kante $e \in \delta(S)$ der in S liegende Knoten durch den neuen Knoten s ersetzt.

Wir kontrahieren nun die Knotenmenge $V \setminus W$ in G und ersetzen $V \setminus W$ durch den neuen Knoten u. Den so entstehenden neuen Graphen bezeichnen wir mit $G' = (W \cup \{u\}, E')$. Wir definieren $y' \in \mathbb{K}^{E'}$ durch $y'_e := y_e \ \forall e \in E(W)$ und $y'_{wu} := \sum_{v \in V \setminus W} y_{wv}$. (Wir nennen y' die *Projektion* von y auf G'.) Aus dieser Konstruktion folgt mühelos, dass der Vektor y' die Restriktionen (i), (ii), (iii) bezüglich G' erfüllt. Da G ein kleinstes Gegenbeispiel ist, folgt daraus, dass $y' \in \mathrm{PMATCH}(G')$ gilt. Und somit ist y' die Konvexkombination von Inzidenzvektoren von perfekten Matchings in G', sagen wir

$$y' = \sum_{M'} \lambda_{M'} \chi^{M'}$$

mit geeigneten $\lambda_{M'} \in [0, 1]$ für alle perfekten Matchings M' in G'.

Analog kontrahieren wir in G nun die Knotenmenge W zu einem neuen Knoten t und erhalten dadurch einen Graphen $G'' = ((V \setminus W) \cup \{t\}, E'')$ und einen Vektor $y'' \in \mathbb{K}^{E''}$. Mit demselben Argument ist y'' enthalten in $\mathrm{PMATCH}(G'')$, und wir können daraus schließen, dass y'' als Konvexkombination von Inzidenzvektoren von perfekten Matchings M'' von G'' dargestellt werden kann:

$$y'' = \sum_{M''} \mu_{M''} \chi^{M''}.$$

Für jedes perfekte Matching M in G mit $\delta(W) = 1$ sei nun M' bzw. M'' das zugehörige perfekte Matching in G' bzw. G'', das durch Kontraktion von $V \setminus W$ bzw. W entsteht. Wir behaupten nun, dass Folgendes gilt:

$$y = \sum_{e \in \delta(W)} \sum_{\substack{M \\ \text{perfektes Matching} \\ \text{mit } M \cap \delta(W) = \{e\}}} \frac{\lambda_{M'} \mu_{M''}}{y_e} \chi^M.$$

Der Nachweis der Korrektheit dieser Formel und dass dadurch y als Konvexkombination von perfekten Matchings in G dargestellt wird, sei dem Leser überlassen. Dieses Ergebnis widerspricht der Annahme, dass y nicht in $\mathrm{PMATCH}(G)$ liegt, und damit ist der Beweis erledigt. □

Satz (3.17) ist der Startpunkt für vielfältige Verallgemeinerungen und Verfeinerungen, von denen wir hier einige (ohne Beweis) angeben wollen. Die Beweise sind nicht wirklich schwierig, aber technisch aufwendig und können z. B. in Part III des dreibändigen Werkes Schrijver (2003) nachgelesen werden. Eine schöne Übersicht über die Polyedertheorie des Matching-Problems gibt der Aufsatz Pulleyblank (2012).

(3.18) Satz. *Für einen Graphen $G = (V, E)$ ist das folgende System von Ungleichungen eine vollständige Beschreibung des Matching-Polytops* $\mathrm{MATCH}(G)$.

(i) $x(\delta(v)) \leq 1 \quad \forall v \in V$

(ii) $x(E(W)) \leq \frac{1}{2}(|W|-1) \quad \forall W \subseteq V$, $|W|$ *ungerade*

(iii) $x_e \geq 0 \quad \forall e \in E.$ △

Neben (einfachen) Matchings ist man in verschiedenen Anwendungen auch an sogenannten c-kapazitierten b-Matchings interessiert. Diese sind wie folgt definiert. Wir starten mit einem Graphen $G = (V, E)$. Für jeden Knoten $v \in V$ ist ein Knotengrad $b_v \in \mathbb{Z}_+$ und für jede Kante $e \in E$ eine Kapazität $c_e \in \mathbb{Z}_+$ gegeben. Wir suchen einen ganzzahligen Vektor $x \in \mathbb{K}^E$, bei dem der Wert x_e die Vielfachheit der Wahl der Kante $e \in E$ beschreibt. Jeder Knoten v darf auf höchstens b_v Kanten (die mit ihrer Vielfachheit gezählt werden) liegen, und jede Kante e darf höchstens c_e Mal gewählt werden. Ein Vektor $x \in \mathbb{Z}_+^E$, der diese Eigenschaft hat, heißt *c-kapazitiertes b-Matching*.

(3.19) Satz. *Sei $G = (V, E)$ ein Graph mit einer Knotengradfunktion $b : V \to \mathbb{Z}_+$ und einer Kantenkapazitätsfunktion $c : E \to \mathbb{Z}_+$. Die konvexe Hülle der c-kapazitierten b-Matchings von G wird vollständig beschrieben durch das folgende System von Ungleichungen:*

(i) $x(\delta(v)) \leq b_v \quad \forall v \in V$

(ii) $x(E(W))+x(F) \leq \lfloor \frac{1}{2}(b(W)+c(F)) \rfloor \quad \forall W \subseteq V$ *und* $F \subseteq \delta(W)$, *so dass* $b(W)+c(F)$ *ungerade ist*

(iii) $0 \leq x_e \leq c_e \quad \forall e \in E.$ △

Ist eine lineare Zielfunktion gegeben, so kann man aufgrund von Satz (3.19) ein optimales c-kapazitiertes b-Matching von G mit Hilfe des durch die Zielfunktion und die Ungleichungen (3.19)(i),(ii),(iii) gegebenen linearen Programms bestimmen. Verlangt man in (i) Gleichheit, so spricht man von *perfekten c-kapazitierten b-Matchings*. Die konvexe Hülle der perfekten c-kapazitierten b-Matchings ist natürlich durch (3.19)(i),(ii),(iii) gegeben, wenn in (i) Gleichheit $x(\delta(v)) = b_v$ gefordert wird.

Ein spezielles perfektes c-kapazitiertes b-Matching ist das *perfekte 2-Matching*, welches nichts anderes als eine Kantenmenge $M \subseteq E$ ist, bei der jeder Knoten auf genau zwei Kanten liegt. Ein perfektes 2-Matching ist die disjunkte Vereinigung von Kreisen, wobei jeder Knoten auf genau einem Kreis liegt. Satz (3.19), spezialisiert auf diesen Fall, liefert dann das folgende Ergebnis.

(3.20) Satz. *Die konvexe Hülle aller Inzidenzvektoren von perfekten 2-Matchings eines Graphen $G = (V, E)$ ist vollständig beschrieben durch:*

(i) $x(\delta(v)) = 2 \quad \forall v \in V$

(ii) $x(E(W)) + x(T) \leq |W| + (|T|-1)/2 \quad \forall W \subseteq V$, $T \subseteq \delta(W)$ *mit* $|T|$ *ungerade*

(iii) $0 \leq x_e \leq 1 \quad \forall e \in E.$ △

Die Ungleichungen (3.20)(ii) werden *2-Matching-Ungleichungen* genannt.

3.4 Travelling-Salesman-Polyeder

Eines der in der kombinatorischen Optimierung am intensivsten untersuchten Probleme ist das Travelling-Salesman-Problem. Wir wollen uns in diesem Abschnitt hauptsächlich mit dem *symmetrischen Travelling-Salesman-Problem* (ab jetzt kurz *TSP*) beschäftigen und das *asymmetrische Travelling-Salesman-Problem* (*ATSP*) nur kurz am Ende erwähnen. Gegeben ist hierbei der vollständige Graph $K_n = (V, E)$, $n \geq 3$, mit „Kantenlängen“ c_e $\forall e \in E$, und gesucht ist ein hamiltonscher Kreis (ab jetzt kurz *Tour* genannt), dessen Länge (= Summe der Längen aller Kanten in der Tour) so klein wie möglich ist.

In einem für die Entwicklung der kombinatorischen Optimierung enorm wichtigen Artikel (siehe hierzu Grötschel und Nemhauser (2008)) haben Dantzig et al. (1954) das TSP als ganzzahliges lineares Programm formuliert und ein 49-Städte TSP optimal gelöst. Dabei haben sie die Grundzüge des bis heute bevorzugten Ansatzes zur Lösung von (schweren) kombinatorischen Optimierungsproblemen skizziert, als da sind: Preprocessing, Variablenfixierung, Branch-and-Bound, Schnittebenen. Dantzig, Fulkerson und Johnson betrachteten das folgende lineare Programm

$$\begin{array}{lll} & \min\ c^T x & \\ \text{(i)} & x(\delta(v)) = 2 & \forall v \in V \\ \text{(ii)} & x(E(W)) \leq |W| - 1 & \forall W \subseteq V,\ 1 \leq |W| \leq |V| - 1 \\ \text{(iii)} & 0 \leq x_e \leq 1 & \forall e \in E \end{array} \tag{3.21}$$

und stellten fest, dass die ganzzahligen Lösungen dieses LPs genau die Inzidenzvektoren der Touren in K_n sind. Die Ungleichungen (i) sind bereits bekannte „Gradgleichungen“ (in jede Stadt fährt man hinein und wieder heraus, also Grad = 2), während die Ungleichungen (ii) *Kurzzyklusbedingungen* (englisch *Subtour Elimination Constraints*) genannt werden, da diese offensichtlich verhindern, dass die Kantenmenge $\{e \in E \mid x_e = 1\}$ einen Kreis enthält. Denn ist (W, C) ein Kreis in K_n, $W \neq V$, so wird die Ungleichung $x(E(W)) \leq |W| - 1$ durch den Inzidenzvektor χ^C verletzt. Erstaunlicherweise gab es bis in die 1980er Jahre Bücher im Operations Research, die vorschlugen statt (3.21) das LP zu lösen, bei dem die Ungleichungen (ii) durch

$$x(C) \leq |C| - 1 \qquad \forall \text{ Kreise } C \subseteq E,\ |C| < n \tag{3.22}$$

ersetzt werden. Ein Argument war z. B., dass jede der Ungleichungen von (3.22) weniger Nichtnullen als die Ungleichungen (ii) von (3.21) enthalten. Das ist eine richtige Beobachtung, aber sie geht völlig an einem ganz wichtigen Aspekt vorbei. Warum sind die Kurzzyklusbedingungen (3.21)(ii) „besser“ als die Kreisbedingungen (3.22)?

Statt der Formulierung des TSPs als ganzzahliges Programm hätte man natürlich lieber eine vollständige Beschreibung des *(symmetrischen) Travelling-Salesman-Polytops*

$$Q_T^n := \text{conv}\{\chi^T \in \mathbb{K}^E \mid T \subseteq E \text{ Tour in } K_n\}. \tag{3.23}$$

Seit den 1970er Jahren haben sich viele diskrete Optimierer mit der Untersuchung dieses Polytops beschäftigt. Übersichtsartikel hierzu sind z. B. Grötschel und Padberg (1985),

mehrere Kapitel in dem Sammelband Gutin und Punnen (2002), sowie die „polyedrischen Kapitel“ in dem Buch Applegate et al. (2006). Es ist unmöglich, alle Ergebnisse hier zu skizzieren. Daher erfolgt nur eine ganz kurze Zusammenfassung von einigen Resultaten, die für die praktische Lösung von TSPs (mit Hilfe von Schnittebenen- und Branch-and-Bound-Verfahren) wichtig sind.

Zunächst einmal können wir die Touren in einem vollständigen Graphen als die Basen eines Unabhängigkeitssystems (alle Kantenmengen, die Teilmengen einer Tour sind) auffassen. (In den 1950er Jahren war dieser Ansatz noch nicht bekannt.) Dann sind die Kurzzyklusbedingungen (3.21)(ii) nichts anderes als Rangungleichungen. Und in der Tat sind dies genau die Ungleichungen, die das Polytop der Wälder in K_n, siehe (3.7)(a), vollständig und irredundant beschreiben – bis auf eine Ausnahme. Es fehlt die Ungleichung $x(E) \leq n-1$.

Man kann also durchaus sagen, dass das TSP-IP aus dem Polytop der Wälder dadurch entsteht, dass man eine Ungleichung weglässt und die Gradgleichung (3.21)(i) hinzufügt. Die Ganzzahligkeit der Lösungsmenge geht dabei allerdings verloren.

Dantzig, Fulkerson und Johnson hatten bereits bemerkt, dass (analog zum perfekten 1-Matching-Problem) die Kurzzyklusbedingungen in Schnittbedingungen umgewandelt werden können. Schreibt man nämlich eine Kurzzyklusbedingung in der skalierten Form $-2x(E(W)) \geq -2|W| + 2$ und addiert man zu dieser die $|W|$ Gleichungen $x(\delta(v)) = 2$ $\forall v \in W$, so erhält man die äquivalente Ungleichung

$$x(\delta(W)) \geq 2,$$

d. h. diese Ungleichung definiert dieselbe Seitenfläche von Q_T^n wie $x(E(W)) \leq |W| - 1$.

Schaut man die Fälle genau an, dann sieht man sofort, dass man sich auf die Knotenmenge $W \subseteq V$, $3 \leq |W| \leq |V| - 3$ beschränken kann. Daraus folgt, dass das Ungleichungssystem (3.21)(ii) äquivalent ersetzt werden kann durch das System

$$x(\delta(W)) \geq 2 \qquad \forall W \subseteq V,\ 3 \leq |W| \leq |V| - 3, \tag{3.24}$$

welches die Schnittversionen der Kurzzyklusbedingungen sind (und die damit sehr ähnlich zu den ungeraden Schnittungleichungen (3.16) für das perfekte Matching-Polytop sind).

Nun überlegen wir uns, dass jede Tour ja auch ein perfektes 2-Matching ist. Folglich sind auch die 2-Matching-Ungleichungen (3.20)(ii) gültig für Q_T^n. Dass die Kurzzyklusbedingungen und 2-Matching-Ungleichungen nicht nur irgendwelche gültigen Ungleichungen sind, sondern dass sie so gut wie alle zur vollständigen Beschreibung von Q_T^n notwendig sind, zeigt der folgende Satz aus Grötschel (1977).

(3.25) Satz. *Seien $n \geq 3$ und $K_n = (V, E)$ der vollständige Graph mit n Knoten und Q_T^n das zugehörige symmetrische Travelling-Salesman-Polytop.*

(a) $\operatorname{aff}(Q_T^n) = \{x \in \mathbb{K}^n \mid x(\delta(v)) = 2\ \forall v \in V\}$.

(b) $\dim Q_T^n = |E| - |V| = n(n-3)/2$.

(c) *Die Ungleichung $x_e \leq 1$ definiert eine Facette von Q_T^n für alle $n \geq 4$.*

(d) *Die Ungleichung* $x_e \geq 0$ *definiert eine Facette von* Q_T^n *für alle* $n \geq 5$.

(e) Kurzzyklusbedingungen: *Für* $n \geq 6$ *und* $W \subseteq V$, $3 \leq |W| \leq |V| - 3$ *definiert jede der drei folgendenden Ungleichungen*

$$x(\delta(W)) \geq 2\,, \quad x(E(W)) \leq |W| - 1\,, \quad x(E(V \setminus W)) \leq |V \setminus W| - 1$$

eine Facette von Q_T^n. *Diese drei Ungleichungen sind äquivalent, d. h. sie definieren alle dieselbe Facette von* Q_T^n.

(f) 2-Matching-Ungleichungen: *Für* $n \geq 6$, $H \subseteq V$, $|H| \geq 3$, $T \subseteq \delta(H)$, $|T| \geq 3$ *und ungerade, so dass je zwei Kanten in* T *keinen gemeinsamen Endknoten haben, definiert die zugehörige* 2*-Matching-Ungleichung*

$$x(E(H)) + x(T) \leq |H| + (|T| - 1)/2$$

eine Facette von Q_T^n. *Dieselbe Facette wird auch von der* 2*-Matching-Ungleichung*

$$x(E(V \setminus H)) + x(T) \leq |V \setminus H| + (|T| - 1)/2$$

definiert. △

Die 2-Matching-Ungleichungen sind auf vielfältige Weise verallgemeinert worden. Wir skizzieren hier die Cliquenbaum-Ungleichungen aus Grötschel und Pulleyblank (1986).

(3.26) Definition. *Seien* $n \geq 6$ *und* $K_n = (V, E)$ *der vollständige Graph auf* n *Knoten. Ein* Cliquenbaum C *ist ein zusammenhängender Untergraph von* K_n, *der aus der Vereinigung von Cliquen (d. h. vollständigen induzierten Untergraphen von* K_n*) besteht, die die folgenden Eigenschaften besitzen:*

(1) *Die Cliquen bestehen aus zwei Sorten, den* Griffen $H_1, \ldots, H_r$ *sowie den* Zinken $T_1, \ldots, T_s$ *(*$H_i \subseteq V$, $T_j \subseteq V$*).*

(2) $T_i \cap T_j = \emptyset$, $1 \leq i < j \leq s$.

(3) $H_i \cap H_j = \emptyset$, $1 \leq i < j \leq r$.

(4) *Jede Zinke enthält mindestens* 2 *und höchstens* $n - 2$ *Knoten und mindestens einen Knoten, der in keinem Griff ist.*

(5) *Jeder Griff hat einen nichtleeren Durchschnitt mit einer ungeraden Anzahl von mindestens drei Zinken.*

(6) *Falls eine Zinke* T *und ein Griff* H *einen nichtleeren Durchschnitt haben, dann ist der Graph, den man durch Entfernung von* $H \cap T$ *aus dem Cliquenbaum* C *erhält, unzusammenhängend.* △

Die etwas merkwürdige Terminologie (Griffe, Zinken) kommt daher, dass, bevor die Cliquenbäume „entdeckt“ wurden, sogenannte Kammungleichungen als Verallgemeinerung von 2-Matching-Ungleichungen untersucht wurden und bei denen die Bezeichnungen Griff (*handle*, daher H) und Zinke (*tooth*, daher T) anschaulich sinnvoll waren.

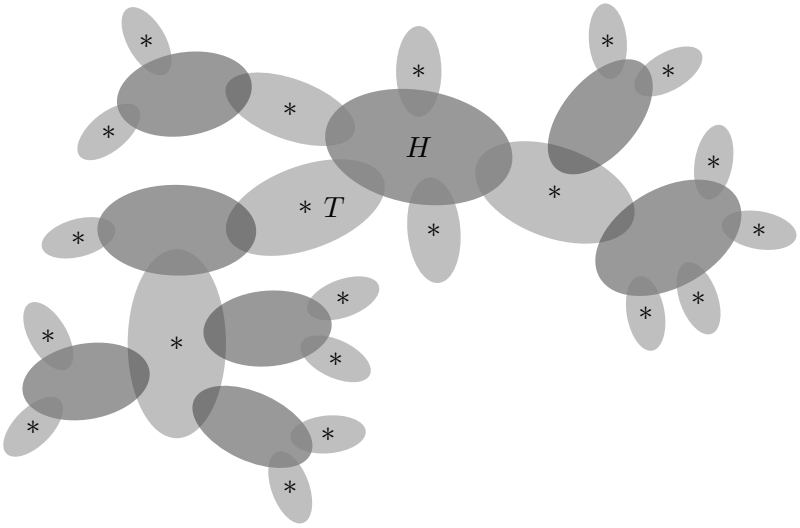


Abbildung 3.2: Ein Cliquenbaum: Griffe sind durch dunkle, Zinken durch hellere Ellipsen angedeutet; jede Zinke muss mindestens einen Knoten außerhalb von Griffen enthalten, angedeutet durch ein $*$

(3.27) Satz. *Seien $n \geq 6$ und C ein Cliquenbaum mit Griffen $H_1, \ldots, H_r$ und Zinken $T_1, \ldots, T_s$. Dann definiert die* Cliquenbaumungleichung

$$\sum_{i=1}^{r} x(E(H_i)) + \sum_{j=1}^{s} x(E(T_j)) \leq \sum_{i=1}^{r} |H_i| + \sum_{j=1}^{s} (|T_j| - t_j) - \frac{s+1}{2} \tag{3.28}$$

eine Facette von Q_T^n, wobei t_j die Anzahl der Griffe angibt, die mit der Zinke T_j einen nichtleeren Durchschnitt haben. △

Die 2-Matching-Ungleichungen tauchen als Spezialfall von (3.28) auf. Sie sind die Cliquenbaumungleichungen mit $r = 1$ und $|T_j| = 2$, $j = 1, \ldots, s$; die *Kammungleichungen* sind die Cliquenbaumungleichungen mit $r = 1$.

Mit den (verschiedenen Versionen der) Kurzzyklusbedingungen, den 2-Matching-Ungleichungen, den Kammungleichungen und den Cliquenbaumungleichungen haben wir

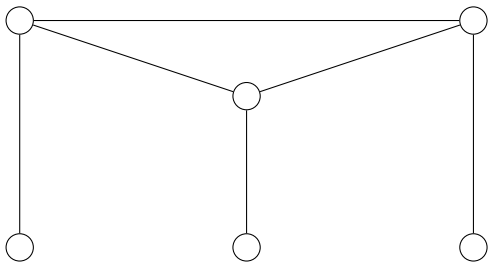


Abbildung 3.3: Ein Kamm (mit $s = 3$ Zinken)

große Klassen von Ungleichungen für das TSP gefunden, die man z. B. in Schnittebenenverfahren einsetzen kann. Vollständige Beschreibungen für Q_T^n sind für alle $n \leq 9$ bekannt. Für $n = 9$ gibt es z. B. nur 20 160 Touren (und damit Ecken von Q_T^n), Q_T^9 hat aber viel mehr Facetten, nämlich 42 104 442. Für $n = 10$ gibt es 181 440 Touren, man kennt 52 043 900 866 verschiedene Facetten. Es ist aber nicht sicher, ob dies alle Facetten von Q_T^{10} sind.

Für das asymmetrische TSP gibt es ähnliche Resultate. Hier ist ein vollständiger Digraph $K_n = (V, A)$, $n \geq 2$ gegeben und die „Bogenlängen“ c_a, $a \in A$ sind nicht notwendigerweise symmetrisch, d. h. es kann $c_{(u,v)} \neq c_{(v,u)}$ für $u, v \in V$, $u \neq v$ gelten. Gesucht ist ein gerichteter Hamilton-Kreis minimaler Gesamtlänge (= Summe der Längen aller Bögen des Kreises). Analog zu (3.21) kann man ein lineares Programm mit Kurzzyklusbedingungen angeben, dessen ganzzahlige Lösungen genau die Inzidenzvektoren der infrage kommenden Kreise sind:

$$\begin{array}{lll} & \min\ c^T x & \\ (\mathrm{i}^+) & x(\delta^+(v)) = 1 & \forall v \in V \\ (\mathrm{i}^-) & x(\delta^-(v)) = 1 & \forall v \in V \\ (\mathrm{ii}) & x(A(W)) \leq |W| - 1 & \forall W \subseteq V,\ 2 \leq |W| \leq |V| - 2 \\ (\mathrm{iii}) & 0 \leq x_a \leq 1 & \forall a \in A \end{array} \tag{3.29}$$

Dies ist aber bei weitem nicht die einzige Möglichkeit der Formulierung. Wählt man einen festen Knoten $v_0 \in V$, führt neue Variablen y_v $\forall v \in V \setminus \{v_0\}$ ein und ersetzt die Kurzzyklusbedingungen (ii) durch die Ungleichungen

$$\begin{array}{ll} y_v - y_u \geq (n-1)x_{(u,v)} - (n-2) & \forall u, v \in V \setminus \{v_0\}, u \neq v \\ 1 \leq y_y \leq n - 1 & \forall v \in V \setminus \{v_0\} \end{array} \tag{3.30}$$

so wird dadurch ein Polyeder

$$P_{x,y} := \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{K}^{|A|+|V|-1} \mid x, y \text{ erfüllen } (3.29)(\mathrm{i}^+), (\mathrm{i}^-), (\mathrm{iii}), (3.30) \right\}$$

definiert. Die Projektion P_x von $P_{x,y}$ auf die ersten $|A|$ Koordinaten enthält dann als ganzzahlige Punkte wieder genau die Inzidenzvektoren aller gerichteten Kreise. Die Ungleichungen (3.30) werden in der Literatur häufig Miller-Tucker-Zemlin-Bedingungen (MTZ) genannt und legen eine „Reihenfolge“ der Knoten fest, wenn man die y-Variablen als ganzzahlige Variablen auffasst (der Knoten v_0 ist dabei der „nullte“ Knoten).

Damit ist es also im Prinzip möglich, die exponentiell vielen Kurzzyklusbedingungen zu umgehen und die gültigen Lösungen mit nur polynomial vielen Ungleichungen zu beschreiben. Allerdings gibt es einen – wie wir später noch sehen werden – schwerwiegenden Nachteil: das Polyeder P_x ist eine echte Obermenge des durch (3.29) definierten Polyeders. Ist z. B. ein Kreis C gegeben, der v_0 nicht enthält, so ergibt eine Summierung der zugehörigen MTZ-Bedingungen:

$$0 = \sum_{(u,v)\in C} y_v - y_u \geq (n-1) \sum_{(u,v)\in C} x_{(u,v)} - |C|(n-2) \implies \sum_{(u,v)\in C} x_{(u,v)} \leq |C| \frac{n-2}{n-1}.$$

Dagegen liefert die zu C gehörige Kurzzyklusbedingung die viel stärkere Einschränkung

$$\sum_{(u,v)\in C} x_{(u,v)} \leq \sum_{(u,v)\in A(C)} x_{(u,v)} \leq |C| - 1$$

Man kann zeigen (siehe z. B. Padberg und Sung (1991)), dass alle Ungleichungen in den x-Variablen, die man aus den MTZ-Bedingungen herleiten kann, solche sind, die wie oben aus Kreisen gewonnen werden, d. h. die Kurzzyklusbedingungen definieren ein kleineres Polyeder als P_x.

Ein weiteres Polyeder für ATSP bekommt man aus folgender Flussformulierung:

$$\begin{aligned} x(\delta^+(v)) &= 1 && \forall v \in V \\ x(\delta^-(v)) &= 1 && \forall v \in V \\ \sum_{(u,v)\in A'} f_{(u,v)} - \sum_{(v,u)\in A'} f_{(v,u)} &= 1 && \forall v \in V \setminus \{v_0\} \\ 0 \leq x_a &\leq 1 && \forall a \in A \\ 0 \leq f_a &\leq (n-1)x_a && \forall a \in A', \end{aligned} \tag{3.31}$$

wobei $A' := A \setminus \{(v, v_0) \mid v \in V \setminus \{v_0\}\}$ alle Bögen des Graphen enthält, außer der in den ausgewählten Knoten v_0 eingehenden. Man kann zeigen, dass die Projektion auf die x-Koordinaten des durch (3.31) definierten Polyeders zwar eine echte Teilmenge des MTZ-Polyeders P_x ist, aber immer noch eine echte Obermenge des von (3.29) definierten Polyeders.

Es gibt noch viele Formulierungen für (A)TSP-Polyeder, manche sind besser (in obigem Sinne), manche äquivalent und manche schlechter als das „Original“ 3.29 von Dantzig, Fulkerson und Johnson; einen ausführlichen Überblick gibt der Artikel Öncan et al. (2009). Mehr Informationen zu verschiedenen polyedrischen Formulierungen kombinatorischer Optimierungsprobleme allgemein und deren Projektionen sind z. B. in Conforti et al. (2010) zu finden. Für weitere Ergebnisse zum ATSP verweisen wir auf Grötschel und Padberg (1985).

3.5 Stabile-Mengen-Polyeder

Eine stabile Menge in einem Graphen $G = (V, E)$ ist eine Knotenmenge S mit der Eigenschaft, dass je zwei Knoten $u, v \in S$ nicht benachbart sind. Das *Stabile-Mengen-Polytop* ist dann wie üblich definiert durch

$$\mathrm{STAB}(G) := \mathrm{conv}\{\chi^S \in \mathbb{K}^V \mid S \subseteq V \text{ stabile Menge}\}, \tag{3.32}$$

wobei die Vektoren $\chi^S = (\chi^S_v)_{v\in V}$ Inzidenzvektoren von stabilen Mengen S sind. Eine LP-Relaxierung von $\mathrm{STAB}(G)$ zu finden, ist ganz einfach:

$$\begin{aligned} &\text{(i)} && x_v \geq 0 && \forall v \in V \\ &\text{(ii)} && x_u + x_v \leq 1 && \forall uv \in E \end{aligned} \tag{3.33}$$

Offensichtlich erfüllt jeder Inzidenzvektor einer stabilen Menge (3.33); und ist $W \subseteq V$ nicht stabil, so gibt es in $E(W)$ eine Kante $e = uv$, und dann erfüllt der Inzidenzvektor χ^W die Ungleichung (3.33)(ii) für diese Kante e nicht.

Die linke Seite des Ungleichungssystems (3.33)(ii) ist die Kanten-Knoten-Inzidenzmatrix des Graphen G. Aus Beispiel (13.11) des Manuskripts von ADM I wissen wir, dass diese Matrix total unimodular ist, wenn G bipartit ist. Und der Satz von Hoffman und Kruskal (ADM I, Satz (13.9)) liefert dann sofort:

(3.34) Satz. *Sei $G = (V, E)$ ein bipartiter Graph, dann gilt*

$$\mathrm{STAB}(G) = \{x \in \mathbb{K}^V \mid x \textit{ erfüllt } (3.33)\}. \qquad \triangle$$

Schon der kleinste nicht bipartite Graph, der K_3, zeigt, dass (3.34) für nicht bipartite Graphen falsch ist. Ordnen wir jedem Knoten $v \in V$ den Wert $x_v = 1/2$ zu, so erfüllt der so definierte Vektor $x \in \mathbb{K}^3$ das System (3.33). Aber offensichtlich ist x keine Konvexkombination der Inzidenzvektoren der vier stabilen Mengen des K_3.

In der Literatur findet man umfangreiche Untersuchungen der Facettenstruktur von $\mathrm{STAB}(G)$. Hier wollen wir nur auf zwei Klassen von Ungleichungen hinweisen, die für $\mathrm{STAB}(G)$ gültig sind. Beide Klassen sind Rangungleichungen bzgl. des Unabhängigkeitssystems der stabilen Mengen auf der Knotenmenge V. Jeder Inzidenzvektor einer stabilen Menge erfüllt das folgende System von Ungleichungen (genannt *Cliquenungleichungen*):

$$x(Q) \leq 1 \qquad \forall \text{ Cliquen } Q \subseteq V, \tag{3.35}$$

denn eine stabile Menge und eine Clique haben höchstens einen gemeinsamen Knoten. Da für jede Kante uv die beiden Knoten u und v eine Clique bilden, bilden die Ungleichungen (3.33)(ii) eine Unterklasse von (3.35). Ein Graph heißt *perfekt*, falls

$$\mathrm{STAB}(G) = \{x \in \mathbb{K}^V \mid x \text{ erfüllt } (3.35), x \geq 0\}.$$

Die Klasse der perfekten Graphen ist eine viel untersuchte Klasse mit sehr interessanten Eigenschaften. Für perfekte Graphen G gilt z. B., dass die maximale Größe einer Clique in G, bezeichnet mit $\omega(G)$, gleich der Färbungszahl $\chi(G)$ ist, und das gilt auch für alle induzierten Untergraphen von G.

Eine zweite Klasse von Ungleichungen kann man aus den für bipartite Graphen verbotenen Untergraphen ableiten: den ungeraden Kreisen. Ist $V(C)$ die Knotenmenge eines ungeraden Kreises in G, so kann eine stabile Menge S offensichtlich höchstens $(|V(C)|-1)/2$ Knoten gemeinsam mit $V(C)$ haben. Diese einfache Überlegung ergibt, dass die Inzidenzvektoren von stabilen Mengen folgendes System von Ungleichungen (genannt *Ungerade-Kreis-Ungleichungen*) erfüllen:

$$x(V(C)) \leq (|V(C)| - 1)/2 \qquad \text{für alle ungeraden Kreise } (V(C), C) \text{ in } G. \tag{3.36}$$

Ein Graph heißt t-perfekt, falls

$$\mathrm{STAB}(G) = \{x \in \mathbb{K}^V \mid x \text{ erfüllt } (3.33) \text{ und } (3.36)\}.$$

Eine ausführliche Darstellung von polyedrischen Aspekten des Stabile-Mengen-Problems, von perfekten und t-perfekten Graphen und Algorithmen zur Lösung des Problems kann man in Kapitel 9 von Grötschel et al. (1993) finden.

Zusammenfassend, für jeden Graphen $G = (V, E)$ definiert das folgende System von Ungleichungen

$$\begin{aligned} x_v &\geq 0 && \forall v \in V \\ x_u + x_v &\leq 1 && \forall uv \in E \\ x(Q) &\leq 1 && \forall \text{ Cliquen } Q \subseteq V \\ x(V(C)) &\leq (|V(C)| - 1)/2 && \forall \text{ ungeraden Kreise } (V(C), C) \text{ in } G \end{aligned} \tag{3.37}$$

ein Polyeder, das STAB(G) enthält und dessen ganzzahlige Lösungen Inzidenzvektoren von stabilen Mengen sind. Eine Cliquenungleichung $x(Q) \leq 1$ definiert eine Facette von STAB(G), falls Q maximal ist (d. h. in keiner weiteren Clique enthalten ist). Auch die ungeraden Kreisungleichungen (3.36), die keine Diagonale enthalten, definieren (unter gewissen Zusatzannahmen) Facetten von STAB(G).

3.6 Separation

Wenn man die in den Abschnitten 3.2 – 3.5 untersuchten kombinatorischen Optimierungsprobleme polyedrisch darstellt, so führt das in fast allen Fällen zu Ungleichungssystemen, deren Größe exponentiell in $|V| + |E|$ wächst. Es ist also unmöglich, diese Ungleichungssysteme explizit in Lösungsverfahren für lineare Programme zu verwenden. Das Besondere an den vorher vorgestellten Ungleichungssystemen ist jedoch, dass die meisten dieser Systeme in polynomialer Zeit „überprüft“ werden können.

(3.38) Definition. *Seien $A \in \mathbb{K}^{(m,n)}$, $b \in K^m$. Die Aufgabe festzustellen, ob der Vektor $y \in K^n$ alle Ungleichungen $Ax \leq b$ erfüllt und in dem Falle, dass y dies nicht tut, eine Ungleichung des Systems zu liefern, die von y verletzt wird, heißt* Separationsproblem (für $Ax \leq b$). △

Für explizit gegebene Systeme $Ax \leq b$ ist Separation trivial. Wir setzen y einfach in das System ein und können so feststellen, ob y alle Ungleichungen erfüllt; und falls dies nicht der Fall ist, liefert die „Einsetzmethode“ direkt eine Ungleichung mit $A_i.y > b_i$.

Das Separationsproblem kann man jedoch manchmal auch für Systeme lösen, die exponentiell viele Ungleichungen enthalten. Betrachen wir z. B. die Kurzzyklusbedingungen für $K_n = (V, E)$ in ihrer Schnittversion (3.24)

$$x(\delta(W)) \geq 2 \qquad \forall W \subseteq V,\ 1 \leq |W| \leq |V| - 1$$

zusammen mit den Gradgleichungen

$$x(\delta(v)) = 2 \qquad \forall v \in V$$

und den Variablenbeschränkungen

$$0 \leq x_e \leq 1 \qquad \forall e \in E.$$

Ist $y \in \mathbb{K}^E$, so können wir durch Einsetzen in die Gradgleichung und Variablenschranken prüfen, ob y diese erfüllt. Falls nicht, haben wir das Separationsproblem gelöst. Falls diese Restriktionen erfüllt sind, dann definieren wir einen Graphen $G = (V, F)$ durch

$$F := \{e \in E \mid y_e > 0\}.$$

Wir betrachten y_e als Kapazität der Kante $e \in F$. Für je zwei Knoten $u, v \in V$ bestimmen wir nun einen u und v trennenden Schnitt $\delta(W)$ minimaler Kapazität $y(\delta(W))$. Dies können wir mit irgendeinem der (vielen) polynomialen Max-Flow-Algorithmen machen. Haben alle minimalen Schnitte eine Kapazität größer oder gleich 2, so erfüllt y offenbar alle Kurzzyklusbedingungen. Jeder Schnitt $\delta(W)$ mit $y(\delta(W)) < 2$ liefert eine Kurzzyklusbedingung oder eine Schnittversion der Kurzzyklusbedingung, die von y verletzt wird. Damit haben wir folgenden Satz bewiesen.

(3.39) Satz. *Das Separationsproblem für die Nebenbedingungen der LP-Relaxierung* (3.21) *des TSPs kann in polynomialer Zeit gelöst werden.* △

Auf ähnliche Weise kann man polynomiale Separationsalgorithmen für die folgenden Klassen von Ungleichungen angeben:

- Rangungleichungen (3.2) für Matroid-Polytope
- Das Ungleichungssystem (3.17) (bzw. (3.18)) (i),(ii),(iii) für das (perfekte) Matching-Polytop.
- Das Ungleichungssystem (3.19)(i),(ii),(iii) für die konvexe Hülle der c-kapazitierten b-Matchings
- Das Ungleichungssystem (3.20)(i),(ii),(iii) für das Polytop der perfekten 2-Matchings
- Die Kurzzyklusbedingungen für das Travelling-Salesman-Polytop
- Die ungeraden Kreis-Ungleichungen (3.36) für das Stabile-Mengen-Polytop.

Bezüglich STAB(G) kann man zeigen, dass die Separation von Cliquenungleichungen $\mathcal{NP}$-schwer ist, man kann aber eine (unendlich große) Obermenge der Cliquenungleichungen in polynomialer Zeit separieren.

Für das TSP gibt es einige Spezialfälle von Kammungleichungen, die in polynomialer Zeit separiert werden können. Für Cliquenbaumungleichungen gibt es nur einige heuristische Separierungsverfahren.

Was hilft das?

Im Verlauf der Vorlesung werden wir sehen, dass Separierungsverfahren sowohl theoretisch als auch praktisch erfolgreich eingesetzt werden können. Das Buch Applegate et al. (2006) zum TSP ist ein eindrucksvolles Beispiel dafür.

Literaturverzeichnis

D. L. Applegate, R. E. Bixby, V. Chvátal, und W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, 2006.

M. Conforti, G. Cornuéjols, und G. Zambelli. Extended formulations in combinatorial optimization. *4OR*, 8:1–48, 2010.

G. B. Dantzig, D. R. Fulkerson, und S. Johnson. Solution of a large-scale traveling salesman problem. *Operations Research*, 2(4):393–410, 1954.

J. Edmonds. Maximum matching and a polyhedron with 0, 1 vertices. *J. Res. Nat. Bureau of Standards*, 6913:125–130, 1965.

M. Grötschel. *Polyedrische Charakterisierungen kombinatorischer Optimierungsprobleme*. Verlag Anton Hain, 1977.

M. Grötschel und G. L. Nemhauser. George Dantzig's contributions to integer programming. *Discrete Optimization*, 5:168–173, 2008.

M. Grötschel und M. W. Padberg. Polyhedral theory. In E. L. Lawler, J. K. Lenstra, A. Rinnooy Kan, und D. Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, S. 251–306. Wiley, 1985.

M. Grötschel und W. R. Pulleyblank. Clique tree inequalities and the symmetric travelling salesman problem. *Mathematics of Operations Research*, 11:537–569, 1986.

M. Grötschel, L. Lovász, und A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1993.

G. Gutin und A. P. Punnen. *The Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, 2002.

T. Öncan, İ. Altınel, und G. Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36:637–654, 2009.

M. Padberg und T. Sung. An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming*, 52:315–357, 1991.

W. R. Pulleyblank. Edmonds, matching and the birth of polyhedral combinatorics. In M. Grötschel, editor, *Optimization Stories*, Documenta Mathematica, S. 181–197. DMV, 2012. Extra Volume ISMP 2012.

A. Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer-Verlag, Berlin, 2003.

4 Die Ellipsoidmethode

In (9.37) im ADM I Skript haben wir angemerkt, dass man Beispiele von Polyedern und Zielfunktionen konstruieren kann, so dass der Simplexalgorithmus alle Ecken der Polyeder durchläuft. Die in den Übungen besprochene Klasse von Klee & Minty-Beispielen, bei denen dieser Fall auftritt, ist definiert durch n Variablen und $2n$ Ungleichungen. Die zugehörigen Polyeder haben 2^n Ecken. Es ist offensichtlich, dass 2^n Iterationen des Simplexalgorithmus zu nicht tolerierbaren Rechenzeiten führen. Man kann zwar zeigen (siehe Borgwardt (1982)), dass das Laufzeitverhalten des Simplexalgorithmus im Durchschnitt gut ist, aber dennoch bleibt die Frage, ob es nicht möglich ist, eine generelle „niedrige“ Schranke für die Anzahl der Iterationsschritte des Simplexalgorithmus (mit einer speziellen Zeilen- und Spaltenauswahlregel) zu finden. Dieses Problem ist bis heute ungelöst!

Im Jahre 1979 hat L. G. Khachiyan (siehe Khachiyan (1979)) gezeigt, dass lineare Programme in „polynomialer Zeit“ gelöst werden können. Das Verfahren, das er dazu angegeben hat, die sogenannte Ellipsoidmethode, arbeitet allerdings völlig anders als der Simplexalgorithmus und scheint in der Praxis im Laufzeitverhalten dem Simplexalgorithmus „im Durchschnitt“ weit unterlegen zu sein. Theoretisch beweisbar ist jedoch, dass es keine Beispiele linearer Programme (z. B. vom Klee & Minty-Typ) gibt, die die Ellipsoidmethode zu exorbitanten Laufzeiten zwingen.

Die dieser Methode zugrundeliegenden Ideen benutzen ganz andere geometrische Überlegungen, als wir sie bisher gemacht haben. Außerdem sind zu einer korrekten Implementation so viele numerische Details zu klären, dass der zeitliche Rahmen dieser Vorlesung durch eine vollständige Diskussion aller Probleme gesprengt würde. Wir wollen daher nur einen Überblick über die Methode geben und nur einige Beweise ausführen.

4.1 Polynomiale Reduktionen

In ADM I haben wir in Kapitel 4 bereits einige Grundbegriffe kennengelernt, die nötig sind, das Laufzeitverhalten eines Algorithmus exakt zu beschreiben. Hier wollen wir die wichtigsten Konzepte dieser Theorie noch einmal für den Spezialfall der linearen Programmierung darstellen.

Zunächst müssen wir unser Konzept, dass Zahlen aus dem Körper $\mathbb{K}$ gewählt werden können, aufgeben. Jede Zahl, die wir in unserem Berechnungsmodell betrachten, muss endlich kodierbar sein. Es gibt aber kein Kodierungsschema, so dass z. B. alle reellen Zahlen durch endlich viele Symbole beschrieben werden könnten. Wir beschränken uns daher auf rationale Zahlen. Haben wir eine Ungleichung

$$a^T x \leq \alpha$$

mit $a \in \mathbb{Q}^n$, $\alpha \in \mathbb{Q}$, so können wir auf einfache Weise das kleinste gemeinsame Vielfache p der Nenner der Komponenten von a und des Nenners von α bestimmen. Die Ungleichung

$$pa^T x \leq p\alpha$$

hat offenbar die gleiche Lösungsmenge wie $a^T x \leq \alpha$, während alle Koeffizienten dieser Ungleichung ganzzahlig sind. Der Fall rationaler Daten lässt sich also direkt auf den Fall ganzzahliger Daten reduzieren. Es ist zwar häufig einfacher unter der Voraussetzung ganzzahliger Daten zu rechnen, und fast alle Aufsätze zur Ellipsoidmethode machen diese Annahme, wir wollen aber dennoch für dieses Kapitel voraussetzen:

(4.1) Annahme. *Alle Daten linearer Programme sind rational, d. h. für jedes LP der Form* $\max c^T x$, $Ax \leq b$ *gilt* $c \in \mathbb{Q}^n$, $A \in \mathbb{Q}^{(m,n)}$, $b \in \mathbb{Q}^m$. △

Wir erinnern noch einmal an die Definition der *Kodierungslänge* aus Abschnitt 4.1 des ADM I Skripts. Für eine ganze Zahl n ist die Kodierungslänge gegeben durch

$$\langle n \rangle := \lceil \log_2(|n| + 1) \rceil + 1$$

und für eine rationale Zahl $r = \frac{p}{q}$ mit p und q teilerfremd und $q > 0$ durch

$$\langle r \rangle := \langle p \rangle + \langle q \rangle.$$

Die Kodierungslänge einer Matrix $A = (a_{ij}) \in \mathbb{Q}^{(m,n)}$ (oder analog eines Vektors) ist

$$\langle A \rangle := \sum_{i=1}^{m} \sum_{j=1}^{n} \langle a_{ij} \rangle.$$

Daraus folgt, dass die Kodierungslänge eines linearen Programms der Form $\max c^T x$, $Ax \leq b$ gegeben ist durch

$$\langle c \rangle + \langle A \rangle + \langle b \rangle.$$

In Abschnitt 4.1 des ADM I Skripts haben wir auch die *Laufzeit* eines Algorithmus' $\mathcal{A}$ zur Lösung eines Problems Π (kurz $L_{\mathcal{A}}(\Pi)$) definiert als die Anzahl der elementaren Rechenschritte, die während der Ausführung des Algorithmus durchgeführt wurden, multipliziert mit der Kodierungslänge der (bezüglich ihrer Kodierungslänge) größten Zahl, die während der Ausführung des Algorithmus aufgetreten ist. Der Algorithmus $\mathcal{A}$ hat dann *polynomiale Laufzeit*, wenn die Laufzeitfunktion nach oben durch ein Polynom $p : \mathbb{N} \to \mathbb{N}$ beschränkt werden kann:

$$f_{\mathcal{A}}(n) \leq p(n) \qquad \forall n \in \mathbb{N},$$

siehe ADM I, Definition (4.2).

Für die Klasse der linearen Programme der Form $\max c^T x$, $Ax \leq b$ muss also die Laufzeit eines Algorithmus durch ein Polynom in $\langle A \rangle + \langle b \rangle + \langle c \rangle$ beschränkt werden können, wenn er polynomial sein soll. Wie die Klee & Minty-Beispiele zeigen, ist der Simplexalgorithmus kein polynomialer Algorithmus zur Lösung linearer Programme!

Die Ellipsoidmethode ist kein Optimierungsverfahren, sondern lediglich eine Methode, die in einem gegebenen Polyeder einen Punkt findet, falls ein solcher existiert. Wir müssen daher das allgemeine lineare Optimierungsproblem auf diesen Fall zurückführen. Aus ADM I, Abschnitt 2.3 (allgemeine Transformationsregeln) wissen wir, dass jedes lineare Programm in der Form

$$\begin{aligned} \max\ & c^T x \\ & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{4.2}$$

geschrieben werden kann. Das zu (4.2) duale Programm ist

$$\begin{aligned} \min\ & b^T y \\ & A^T y \geq c \\ & y \geq 0. \end{aligned} \tag{4.3}$$

Aus dem Dualitätssatz (11.24) im ADM I Skript wissen wir, dass (4.2) und (4.3) genau dann optimale Lösungen haben, deren Werte gleich sind, wenn beide Programme zulässige Lösungen haben. Daraus folgt, dass jedes Element $\binom{x}{y}$ des Polyeders P, definiert durch

$$\begin{pmatrix} -c^T & b^T \\ A & 0 \\ 0 & -A^T \\ -I & 0 \\ 0 & -I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 0 \\ b \\ -c \\ 0 \\ 0 \end{pmatrix} \tag{4.4}$$

eine Optimallösung x von (4.2) und eine Optimallösung y von (4.3) bestimmt. Dies impliziert, dass es zur Lösung von (4.2) genügt, einen Punkt in (4.4) zu finden.

Der obige „Trick“, das Primal- und das Dualprogramm zusammenzufassen, bläht natürlich das zu bearbeitende System ungeheuer auf. Eine andere Methode der Reduktion ist die binäre Suche, die wir nun schildern. Zur korrekten Darstellung benötigen wir jedoch noch einige (auch für andere Zwecke) wichtige Hilfssätze. Zunächst wollen wir einige Parameter durch die Kodierungslänge abschätzen. Um den ersten Hilfssatzes zu beweisen, benötigen wir die bekannte Hadamard-Ungleichung, die besagt, dass das Volumen eines Parallelepipeds in $\mathbb{R}^n$ mit Kantenlängen $d_1, \ldots, d_n$ nicht größer ist als das Volumen des Würfels mit Kantenlängen $d_1, \ldots, d_n$. Bekanntlich ist das Volumen eines Parallelepipeds, das durch die Vektoren $a_1, \ldots, a_n$ aufgespannt wird, nichts anderes als der Absolutbetrag der Determinante der Matrix A mit Spalten $A_{\cdot 1} = a_1, \ldots, A_{\cdot n} = a_n$. Die *Hadamard-Ungleichung* lautet also

$$|\det A| \leq \prod_{j=1}^{n} \|A_{\cdot j}\|_2 \tag{4.5}$$

(4.6) Lemma.

(a) *Für jede Zahl* $r \in \mathbb{Q}$ *gilt:* $|r| \leq 2^{\langle r \rangle - 1} - 1$.

(b) *Für je zwei rationale Zahlen r, s gilt:* $\langle rs\rangle \le \langle r\rangle + \langle s\rangle$.

(c) *Für jeden Vektor $x \in \mathbb{Q}^n$ gilt:* $\|x\|_2 \le \|x\|_1 \le 2^{\langle x\rangle - n} - 1$.

(d) *Für jede Matrix $A \in \mathbb{Q}^{(n,n)}$ gilt:* $|\det(A)| \le 2^{\langle A\rangle - n^2} - 1$. △

Beweis. (a) und (b) folgen direkt aus der Definition.
(c) Sei $x = (x_1, \ldots, x_n)^T \in \mathbb{Q}^n$, dann gilt nach (a)

$$1 + \|x\|_1 \;=\; 1 + \sum_{i=1}^{n} |x_i| \;\le\; \prod_{i=1}^{n} (1 + |x_i|) \;\le\; \prod_{i=1}^{n} 2^{\langle x_i\rangle - 1} \;=\; 2^{\langle x\rangle - n}.$$

Die Ungleichung $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2} \le \sum_{i=1}^n |x_i| = \|x\|_1$ ist trivial.
(d) Aus der Hadamard-Ungleichung (4.5) und (c) folgt

$$1 + |\det(A)| \;\le\; 1 + \prod_{j=1}^{n} \|A_{\cdot j}\|_2 \;\le\; \prod_{j=1}^{n} (1 + \|A_{\cdot j}\|_2) \;\le\; \prod_{j=1}^{n} 2^{\langle A_{\cdot j}\rangle - n} \;=\; 2^{\langle A\rangle - n^2}.$$

□

Diesen Hilfssatz können wir zur Abschätzung der „Größe“ der Ecken von Polyedern wie folgt benutzen.

(4.7) Satz. *Für jede Ecke $v = (v_1, \ldots, v_n)^T$ eines Polyeders P der Form $P(A, b)$, $P^=(A, b)$ oder $P = \{x \in \mathbb{R}^n \mid Ax \le b, x \ge 0\}$, $A \in \mathbb{Q}^{(m,n)}$, $b \in \mathbb{Q}^m$, gilt*

(a) *Der Absolutbetrag des Zählers von v_i ist höchstens $2^{\langle A\rangle + \langle b\rangle - n^2}$, der Absolutbetrag des Nenners von v_i ist höchstens $2^{\langle A\rangle - n^2}$, $i = 1, \ldots, n$.*

(b) $|v_i| \le 2^{2\langle A\rangle + \langle b\rangle - 2n^2}$, $i = 1, \ldots, n$.

(c) *Falls $A \in \mathbb{Z}^{(m,n)}$, so gilt $|v_i| \le 2^{\langle A\rangle + \langle b\rangle - n^2}$, $i = 1, \ldots, n$.* △

Beweis. Ist v Ecke von P, so gibt es nach Satz (8.8) aus dem ADM I Skript eine reguläre Untermatrix und einen entsprechenden Untervektor des Restriktionensystems von P, so dass v die eindeutig bestimmte Lösung des hierdurch bestimmten Gleichungssystems ist. Wir führen den Fall $P = \{x \mid Ax \le b, x \ge 0\}$ vor. Die beiden anderen Fälle verlaufen analog.

Es gibt also eine (n, n)-Matrix D, deren Zeilen Zeilen von A oder negative Einheitsvektoren sind und einen entsprechenden Vektor d, dessen Komponenten Elemente von b oder Nullen sind, so dass v die eindeutige Lösung von $Dx = d$ ist. Nach der Cramerschen Regel gilt dann für $i = 1, \ldots, n$

$$v_i = \frac{\det D_i}{\det D},$$

wobei D_i aus D dadurch entsteht, dass die i-te Spalte von D durch den Vektor d ersetzt wird. Hat D Zeilen, die negative Einheitsvektoren sind, so bezeichnen wir mit $\bar{D}$ die

Matrix, die durch Streichen dieser Zeilen und der Spalten, in denen sich die Elemente -1 befinden, entsteht. Aufgrund des Determinantenentwicklungssatzes gilt $|\det D| = |\det \bar{D}|$, und ferner ist $\bar{D}$ eine Untermatrix von A. Daraus folgt mit (4.6)(d)

$$|\det D| = |\det \bar{D}| \leq 2^{\langle \bar{D} \rangle - n^2} \leq 2^{\langle A \rangle - n^2}.$$

Analog folgt

$$|\det D_i| \leq 2^{\langle A \rangle + \langle b \rangle - n^2}.$$

Damit ist (a) bewiesen.

Ist $|\det D| \geq 1$, dies ist z. B. dann der Fall, wenn $A \in \mathbb{Z}^{(m,n)}$ gilt, so erhalten wir mit (a)

$$|v_i| \leq |\det D_i| \leq 2^{\langle A \rangle + \langle b \rangle - n^2}.$$

Hieraus folgt (c).

Ist $|\det D| < 1$, so müssen wir $|\det D|$ nach unten abschätzen. Sei $q = \prod_{i,j} q_{ij}$ das Produkt der Nenner der Elemente $d_{ij} = \frac{p_{ij}}{q_{ij}}$ von D. Dann gilt $|\det D| = \frac{q|\det D|}{q}$, und sowohl q als auch $q|\det D|$ sind ganze Zahlen. Aus (4.6)(a), (b) folgt

$$q = \prod_{i,j} q_{ij} \leq 2^{\langle \prod_{i,j} q_{ij} \rangle} \leq 2^{\sum_{i,j} \langle q_{ij} \rangle} \leq 2^{\langle D \rangle - n^2} \leq 2^{\langle A \rangle - n^2}.$$

Daraus ergibt sich

$$|v_i| \leq \frac{|\det D_i|}{|\det D|} = \frac{|\det D_i| q}{q|\det D|} \leq |\det D_i| q \leq 2^{2\langle A \rangle + \langle b \rangle - 2n^2}.$$

Damit ist auch (b) bewiesen. □

Da nach ADM I, Satz (8.11) alle Polyeder P der Form $P = \{x \mid Ax \leq b, x \geq 0\}$ spitz sind und nach ADM I, Folgerung (8.14) lineare Programme über spitzen Polyedern optimale Ecklösungen haben (falls sie überhaupt Optimallösungen haben) folgt:

(4.8) Satz. *Das lineare Programm (4.2) hat eine Optimallösung genau dann, wenn die beiden linearen Programme*

$$\begin{aligned} \max \quad & c^T x \\ & Ax \leq b \\ & x \geq 0 \\ & x_i \leq 2^{2\langle A \rangle + \langle b \rangle - 2n^2} \qquad i = 1, \ldots, n \end{aligned} \tag{4.9}$$

$$\begin{aligned} \max \quad & c^T x \\ & Ax \leq b \\ & x \geq 0 \\ & x_i \leq 2^{2\langle A \rangle + \langle b \rangle - 2n^2} + 1 \qquad i = 1, \ldots, n \end{aligned} \tag{4.10}$$

eine Optimallösung haben und die Werte der Optimallösungen übereinstimmen. Die optimalen Zielfunktionswerte von (4.9) und (4.10) stimmen genau dann nicht überein, wenn (4.2) unbeschränkt ist. (4.2) ist genau dann nicht lösbar, wenn (4.9) oder (4.10) nicht lösbar ist. △

Wir haben damit das lineare Programmierungsproblem (4.2) über einem (allgemeinen) Polyeder auf die Lösung zweier linearer Programme über Polytopen reduziert. Wir müssen also im Prinzip nur zeigen, wie man LPs über Polytopen löst.

(4.11) Satz. *Ist $P \neq \emptyset$ ein Polytop der Form $P(A,b)$, $P^{=}(A,b)$ oder $P = \{x \mid Ax \leq b, x \geq 0\}$ mit $A \in \mathbb{Q}^{(m,n)}$, $b \in \mathbb{Q}^m$, so kann für $c \in \mathbb{Q}^n$ das lineare Programm $\max c^T x$, $x \in P$ nur Optimalwerte in der endlichen Menge*

$$S := \{\frac{p}{q} \in \mathbb{Q} \mid |p| \leq n \cdot 2^{\langle A\rangle+\langle b\rangle+2\langle c\rangle-n^2-n},\ 1 \leq q \leq 2^{\langle A\rangle+\langle c\rangle-n^2-n}\}$$

annehmen. △

Beweis. Da alle Optimalwerte Zielfunktionswerte von Ecken von P sind, brauchen wir nur die Werte $c^T v$, v Ecke von P, abzuschätzen. Sei also $v = (v_1, \ldots, v_n)^T$ eine Ecke von P. Wie im Beweis von (4.7) können wir feststellen, dass die Komponenten v_i von v eine Darstellung der Form

$$v_i = \frac{\det D_i}{\det D} =: \frac{p_i}{d}$$

haben, wobei D eine reguläre Untermatrix von A bzw. $\binom{A}{I}$ ist und D_i aus D dadurch hervorgeht, dass die i-te Spalte von D durch einen Untervektor der rechten Seite ersetzt wird. Satz (4.7) zeigt $d \leq 2^{\langle A\rangle-n^2}$, $p_i \leq 2^{\langle A\rangle+\langle b\rangle-n^2}$. Sei nun $c = (\frac{s_1}{t_1}, \ldots, \frac{s_n}{t_n})^T \in \mathbb{Q}^n$ eine teilerfremde Darstellung der Zielfunktion, so gilt

$$c^T v = \sum_{i=1}^{n} \frac{s_i p_i}{t_i d} = \frac{1}{dt} \sum_{i=1}^{n} s_i p_i \bar{t}_i \quad \text{mit} \quad t := t_1 \cdot \cdots \cdot t_n,\ \bar{t}_i := \frac{t}{t_i}.$$

Aus (4.6)(a) folgt $t = t_1 \cdot \cdots \cdot t_n \leq 2^{\langle t_1\rangle-1} \cdot \cdots \cdot 2^{\langle t_n\rangle-1} = 2^{\sum_i(\langle t_i\rangle-1)} \leq 2^{\langle c\rangle-n}$ und somit

$$q := dt \leq 2^{\langle A\rangle+\langle c\rangle-n^2-n}.$$

Analog folgt

$$p := \sum_{i=1}^{n} s_i p_i \bar{t}_i \leq \sum_{i=1}^{n} 2^{\langle s_i\rangle-1} 2^{\langle A\rangle+\langle b\rangle-n^2} 2^{\langle c\rangle-n} \leq n \cdot 2^{\langle A\rangle+\langle b\rangle+2\langle c\rangle-n^2-n}.$$

□

Satz (4.11) gibt uns nun die Möglichkeit, das Verfahren der binären Suche zur Lösung von $\max c^T x$, $x \in P$ anzuwenden. Diese Methode funktioniert bezüglich der in Satz (4.11) definierten Menge S wie folgt.

(4.12) Algorithmus Binäre Suche.

1. Wähle ein Element $s \in S$, so dass für $S' := \{t \in S \mid t < s\}$ und $S'' := \{t \in S \mid t \geq s\}$ gilt
$$|S'| \leq |S''| \leq |S'| + 1.$$

2. Überprüfe, ob das Polyeder
$$P_s = \{x \mid Ax \leq b, x \geq 0, c^T x \geq s\}$$
nicht leer ist.

3. Ist P_s leer, so setze $S := S'$, andernfalls setze $S := S''$.

4. Ist $|S| = 1$, so gilt für $s \in S$: $s = \max\{c^T x \mid Ax \leq b,\ x \geq 0\}$, und jeder Punkt in P_s ist eine Optimallösung von $\max c^T x$, $x \in P$. Andernfalls gehe zu 1. △

Die Korrektheit dieses Verfahrens ist offensichtlich. Ist die Binärsuche auch effizient? Da in jedem Schritt die Kardinalität $|S|$ von S (fast) halbiert wird, ist klar, dass höchstens

$$\overline{\overline{N}} := \lceil \log_2(|S|) \rceil \tag{4.13}$$

Aufspaltungen von S in zwei Teile notwendig sind, um eine einelementige Menge zu erhalten. Also muss Schritt 2 von (4.12) $\overline{\overline{N}}$ mal ausgeführt werden. Nach (4.11) gilt

$$|S| \leq 2n \cdot 2^{2\langle A\rangle + \langle b\rangle + 3\langle c\rangle - 2n^2 - 2n} + 1,$$

und daraus folgt, dass Test 2 von (4.12) höchstens

$$\overline{N} := 2\langle A\rangle + \langle b\rangle + 3\langle c\rangle - 2n^2 - 2n + \log_2 n + 2 \tag{4.14}$$

mal durchgeführt wurde. Ist Test 2 in einer Zeit ausführbar, die polynomial in $\langle A\rangle + \langle b\rangle + \langle c\rangle$ ist, so ist auch die binäre Suche ein polynomialer Algorithmus, da ein polynomialer Algorithmus für (4.12). nur $\overline{N}$ mal, also nur polynomial oft ausgeführt werden muss.

(4.15) Korollar. *Es gibt einen polynomialen Algorithmus zur Lösung linearer Programme genau dann, wenn es einen Algorithmus gibt, der in polynomialer Zeit entscheidet, ob ein Polytop P leer ist oder nicht und der, falls $P \neq \emptyset$, einen Punkt in P findet.* △

Damit haben wir das lineare Programmierungsproblem reduziert auf die Frage der Lösbarkeit von Ungleichungssystemen, deren Lösungsmenge beschränkt ist.

4.2 Beschreibung der Ellipsoidmethode

In diesem Abschnitt setzen wir $n \geq 2$ voraus. Wir wollen zunächst einige Eigenschaften von Ellipsoiden beschreiben. Wir erinnern daran, dass Ellipsoide volldimensionale konvexe Körper im $\mathbb{R}^n$ sind, die eine besonders einfache Darstellung haben. Und zwar ist eine Menge $E \subseteq \mathbb{R}^n$ ein *Ellipsoid (mit Zentrum a)* genau dann, wenn es einen Vektor $a \in \mathbb{R}^n$ und eine (symmetrische) positiv definite Matrix A gibt, so dass gilt

$$E = E(A, a) := \{x \in \mathbb{R}^n \mid (x - a)^T A^{-1}(x - a) \leq 1\} \tag{4.16}$$

Aus der linearen Algebra wissen wir, dass für symmetrische Matrizen $A \in \mathbb{R}^{(n,n)}$ die folgenden Aussagen äquivalent sind:

(i) A ist positiv definit.

(ii) A^{-1} ist positiv definit.

(iii) Alle Eigenwerte von A sind positive reelle Zahlen.

(iv) $\det(A_{II}) > 0$ für alle Mengen $I = \{1, \ldots, i\}$, $i = 1, \ldots, n$.

(v) $A = B^T B$ für eine reguläre Matrix $B \in \mathbb{R}^{(n,n)}$.

Das Ellipsoid $E(A, a)$ ist durch die (ebenfalls positiv definite) Inverse A^{-1} von A definiert. Das erscheint zunächst seltsam, liegt aber daran, dass sich viele Eigenschaften von $E(A, a)$ aus algebraischen Eigenschaften von A ableiten lassen. Z. B. ist der Durchmesser (d. h. die Länge der längsten Achse) von $E(A, a)$ gleich $2\sqrt{\Lambda}$, wobei Λ der größte Eigenwert von A ist. Die längsten Achsen sind durch die Eigenvektoren, die zu Λ gehören, gegeben. Die Symmetrieachsen von $E(A, a)$ entsprechen ebenfalls den Eigenvektoren von A. In Abbildung 4.1 ist das Ellipsoid $E(A, 0)$ dargestellt mit

$$A = \begin{pmatrix} 16 & 0 \\ 0 & 4 \end{pmatrix}.$$

Die Eigenwerte von A sind $\Lambda = 16$ und $\lambda = 4$ mit den zugehörigen Eigenvektoren $e_1 = (1, 0)^T$ und $e_2 = (0, 1)^T$. Der Durchmesser von $E(A, 0)$ ist $2\sqrt{\Lambda} = 8$.

Zu jeder positiv definiten Matrix A gibt es eine eindeutig bestimmte positive definite Matrix, die mit $A^{\frac{1}{2}}$ bezeichnet und *Wurzel von A* genannt wird, mit

$$A = A^{\frac{1}{2}} A^{\frac{1}{2}}.$$

Die Einheitskugel $S(0, 1) \subseteq \mathbb{R}^n$ (um den Nullpunkt mit Radius 1) ist das Ellipsoid $E(I, 0)$. Man rechnet leicht nach, dass gilt:

$$E(A, a) = A^{\frac{1}{2}} S(0, 1) + a.$$

Damit sei die Aufzählung von Eigenschaften von Ellipsoiden beendet.

Wir wollen nun die geometrische Idee, die hinter der Ellipsoidmethode steckt, erläutern.

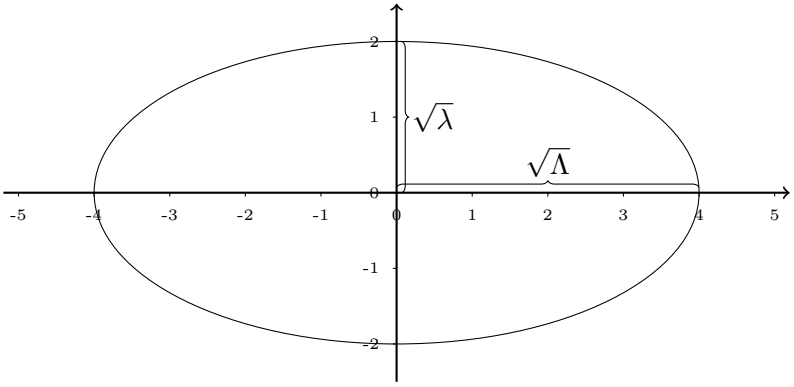


Abbildung 4.1: Ellipsoid mit Symmetrieachsen der Längen $2\sqrt{\Lambda} = 8$ und $2\sqrt{\lambda} = 4$

(4.17) Algorithmus (Geometrische Beschreibung der Ellipsoidmethode). Gegeben sei ein Polytop P. Wir wollen einen Punkt in P finden oder beweisen, dass P leer ist.

1. Konstruiere ein Ellipsoid $E_0 = E(A_0, a_0)$, das P enthält. Setze $k := 0$.
2. Ist das gegenwärtige Ellipsoid E_k „zu klein“, so brich ab mit der Antwort „P ist leer“. STOP.
3. Teste ob der Mittelpunkt a_k von E_k in P enthalten ist.
4. Gilt $a_k \in P$, dann haben wir unser Ziel erreicht, STOP.
5. Gilt $a_k \notin P$, dann gibt es eine P definierende Ungleichung, sagen wir

   $$c^T x \leq \gamma,$$

   die von a_k verletzt wird, d. h. $c^T a_k > \gamma$. Mit

   $$E'_k := E_k \cap \{x \mid c^T x \leq c^T a_k\}$$

   bezeichnen wir das Halbellipsoid von E_k, das P enthält. Wir konstruieren nun das Ellipsoid kleinsten Volumens, das E'_k enthält und nennen es E_{k+1}.
6. Setze $k := k + 1$ und gehe zu 2. △

Das Prinzip des Verfahrens ist klar. Man zäunt das Polytop P durch ein Ellipsoid E_k ein. Ist das Zentrum von E_k nicht in P, so sucht man sich ein kleineres Ellipsoid E_{k+1}, das P enthält und fährt so fort. Die Probleme, die zu klären sind, sind die folgenden:

- Wie findet man ein Ellipsoid, das P enthält?
- Wie kann man E_{k+1} aus E_k konstruieren?

- Wann kann man abbrechen, d. h. was heißt „zu klein“?
- Wieviele Iterationen des Verfahrens sind durchzuführen?

Die nachfolgenden Sätze beantworten diese Fragen, wobei wir nur einige der Beweise, die zum Nachweis der Polynomialität des Verfahrens notwendig sind, angeben wollen.

Unser Anfangsellipsoid soll eine Kugel mit dem Nullpunkt als Zentrum sein. Enthalten die Restriktionen, die das Polytop P definieren, explizite obere und untere Schranken für die Variablen, sagen wir

$$l_i \leq x_i \leq u_i \qquad i = 1, \ldots, n$$

so sei

$$R := \sqrt{\sum_{i=1}^{n} (\max\{|u_i|, |l_i|\})^2}. \tag{4.18}$$

Dann gilt trivialerweise $P \subseteq S(0, R) = E(R^2 I, 0)$. Andernfalls kann man zeigen

(4.19) Lemma. *Sei P ein Polyeder der Form $P(A, b)$, $P^{=}(A, b)$ oder $\{x \in \mathbb{R}^n \mid Ax \leq b,\ x \geq 0\}$ mit $A \in \mathbb{Q}^{(m,n)}$, $b \in \mathbb{Q}^m$, dann gilt*

(a) *Alle Ecken von P sind in der Kugel $S(0, R)$ enthalten mit*

$$R := \sqrt{n} 2^{2\langle A\rangle + \langle b\rangle - 2n^2}.$$

(b) *Ist P ein Polytop, so gilt $P \subseteq S(0, R) = E(R^2 I, 0)$.* △

Beweis. Nach Satz (4.7) gilt für jede Ecke $v^T = (v_1, \ldots, v_n)$ von P

$$|v_i| \leq 2^{2\langle A\rangle + \langle b\rangle - 2n^2} \qquad (i = 1, \ldots, n),$$

und daraus folgt für die euklidische Norm von v:

$$\|v\| \;=\; \sqrt{\sum_{i=1}^{n} v_i^2} \;\leq\; \sqrt{n \max\{v_i^2\}} \;\leq\; \sqrt{n}\, 2^{2\langle A\rangle + \langle b\rangle - 2n^2}.$$

Also ist jede Ecke von P in $S(0, R)$ enthalten. Ist insbesondere P ein Polytop, so folgt daraus $P \subseteq S(0, R)$. □

Damit haben wir durch (4.18) oder (4.19) ein Anfangsellipsoid E_0 gefunden, mit dem wir die Ellipsoidmethode beginnen können. Die Konstruktion von E_{k+1} aus E_k geschieht wie folgt.

(4.20) Satz. *Sei $E_k = E(A_k, a_k) \subseteq \mathbb{R}^n$ ein Ellipsoid, $c \in \mathbb{R}^n \setminus \{0\}$ und $E'_k := \{x \in \mathbb{R}^n \mid c^T x \leq c^T a_k\} \cap E_k$. Setze*

$$d := \frac{1}{\sqrt{c^T A_k c}} A_k c, \tag{4.21}$$

$$a_{k+1} := a_k - \frac{1}{n+1} d, \tag{4.22}$$

$$A_{k+1} := \frac{n^2}{n^2-1}(A_k - \frac{2}{n+1} dd^T), \tag{4.23}$$

dann ist A_{k+1} positiv definit und $E_{k+1} := E(A_{k+1}, a_{k+1})$ ist das eindeutig bestimmte Ellipsoid minimalen Volumens, das E'_k enthält. △

Beweis. Siehe Bland et al. (1981) oder Grötschel et al. (1988). □

Wir wollen kurz den geometrischen Gehalt der Schritte in Satz (4.20) erläutern.

Ist $c \in \mathbb{R}^n$, und wollen wir das Maximum oder Minimum von $c^T x$ über E_k finden, so kann man dies explizit wie folgt angeben. Für den in (4.21) definierten Vektor d und

$$z_{\max} := a_k + d, \qquad z_{\min} := a_k - d$$

gilt

$$c^T z_{\max} = \max\{c^T x \mid x \in E_k\} = c^T a_k + \sqrt{c^T A_k c},$$
$$c^T z_{\min} = \min\{c^T x \mid x \in E_k\} = c^T a_k - \sqrt{c^T A_k c}.$$

Diese Beziehung kann man leicht – z. B. aus der Cauchy-Schwarz-Ungleichung – ableiten. Daraus folgt, dass der Mittelpunkt a_{k+1} des neuen Ellipsoids E_{k+1} auf dem Geradenstück zwischen a_k und $z_{\min}$ liegt. Die Länge dieses Geradenstücks ist $\|d\|$, und a_{k+1} erreicht man von a_k aus, indem man einen Schritt der Länge $\frac{1}{n+1}\|d\|$ in Richtung $-d$ macht. Der Durchschnitt des Randes des Ellipsoids E_{k+1} mit dem Rand von E'_k wird gebildet durch den Punkt $z_{\min}$ und $E''_k := \{x \mid (x-a_k)^T A_k^{-1}(x-a_k) = 1\} \cap \{x \mid c^T x = c^T a_k\}$. E''_k ist der Rand eines „$(n-1)$-dimensionalen Ellipsoids“ im $\mathbb{R}^n$.

In Abbildung 4.2 sind das Ellipsoid E_k aus Abbildung 4.1 und das Ellipsoid E_{k+1}, definiert durch Satz (4.20) bezüglich des Vektors $c^T = (-1, -2)$ dargestellt. E'_k ist grau eingezeichnet. Als Anfangsdaten haben wir daher: Die verletzte Ungleichung ist $-x_1 - 2x_2 \leq -4$. Die zugehörige Gerade $\{x \mid -x_1 - 2x_2 = -4\}$ ist ebenfalls gezeichnet.

$$a_k = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad A_k = \begin{pmatrix} 16 & 0 \\ 0 & 4 \end{pmatrix}, \quad c = \begin{pmatrix} -1 \\ -2 \end{pmatrix}.$$

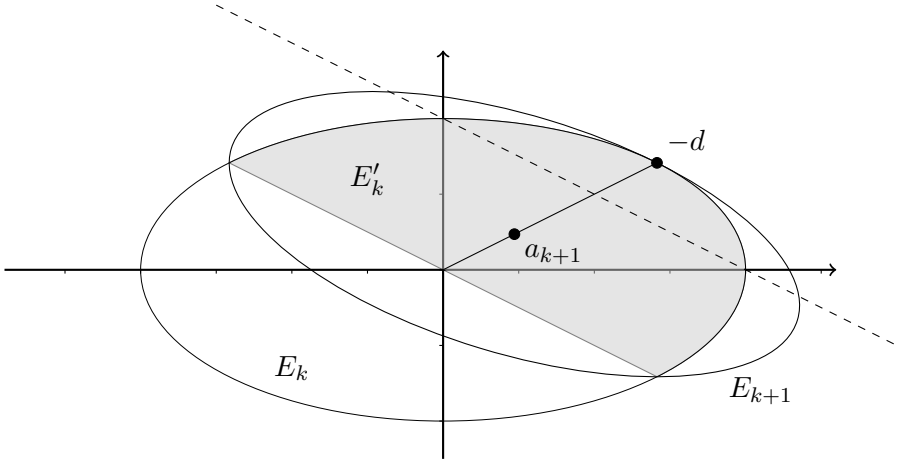


Abbildung 4.2: Ellipsoid E_k aus Abbildung 4.1 und daraus konstruiertes Ellipsoid E_{k+1}

Die Formeln (4.21), (4.22), (4.23) ergeben:

$$\begin{aligned} d &= \frac{-1}{\sqrt{2}}\begin{pmatrix}4\\2\end{pmatrix}, \\ a_{k+1} &= a_k - \frac{1}{3}d = \frac{1}{3\sqrt{2}}\begin{pmatrix}4\\2\end{pmatrix} \approx \begin{pmatrix}0.9428\\0.4714\end{pmatrix}, \\ A_{k+1} &= \frac{n^2}{n^2-1}(A_k - \frac{2}{n+1}dd^T) = \frac{4}{9}\begin{pmatrix}32 & -8\\-8 & 8\end{pmatrix}, \\ E_{k+1} &= E(A_{k+1}, a_{k+1}). \end{aligned}$$

Das Stopkriterium der Ellipsoidmethode beruht auf einem Volumenargument. Nach Konstruktion ist klar, dass das Volumen von E_{k+1} (bezeichnet mit $\mathrm{vol}(E_{k+1})$) kleiner ist als das von E_k. Man kann den Volumenschrumpfungsfaktor explizit berechnen. Er hängt nur von der Dimension n des Raumes $\mathbb{R}^n$ und nicht etwa von Update-Vektor c ab.

(4.24) Lemma.

$$\frac{\mathrm{vol}(E_{k+1})}{\mathrm{vol}(E_k)} \;=\; \left(\left(\frac{n}{n+1}\right)^{n+1}\left(\frac{n}{n-1}\right)^{n-1}\right)^{\frac{1}{2}} \;\leq\; e^{-\frac{1}{2n}} \;<\; 1. \qquad \triangle$$

Beweis. Siehe Grötschel et al. (1988), Lemma (3.1.34). □

Wir sehen also, dass mit den Formeln aus Satz (4.20) eine Folge von Ellipsoiden konstruiert werden kann, so dass jedes Ellipsoid E_{k+1} das Halbellipsoid E'_k und somit das Polytop P enthält und dass die Volumina der Ellipsoide schrumpfen. Die Folge der Volumina konvergiert gegen Null, muss also einmal das Volumen von P, falls P ein positives Volumen hat, unterschreiten. Daher muss nach endlich vielen Schritten der Mittelpunkt eines der Ellipsoide in P sein, falls $P \neq \emptyset$. Wir wollen nun ausrechnen, nach wievielen Schritten dies der Fall ist.

(4.25) Lemma. *Seien* $P = P(A,b) \subseteq \mathbb{R}^n$, $\mathring{P} = \{x \in \mathbb{R}^n \mid Ax < b\}$, *und* $R := \sqrt{n}2^{2\langle A\rangle+\langle b\rangle-2n^2}$.

(a) *Entweder gilt* $\mathring{P} = \emptyset$ *oder*

$$\mathrm{vol}(\mathring{P} \cap S(0,R)) \geq 2^{-(n+1)(\langle A\rangle+\langle b\rangle-n^2)}.$$

(b) *Ist* P *volldimensional (d. h.* $\dim P = n$*), dann gilt*

$$\mathrm{vol}(P \cap S(0,R)) = \mathrm{vol}(\mathring{P} \cap S(0,R)) > 0. \qquad \triangle$$

Lemma (4.25) zusammen mit Lemma (4.19) sind geometrisch und algorithmisch interessant. Die beiden Hilfssätze implizieren folgendes.

- Wenn ein Polyeder P nicht leer ist, dann gibt es Elemente des Polyeders, die „nah“ beim Nullvektor liegen, d. h. in $S(0,R)$ enthalten sind.
- Es reicht aus zu überprüfen, ob $P \cap S(0,R)$ leer ist oder nicht. Daraus kann man schließen, dass P leer ist oder nicht.
- Will man einen Konvergenzbeweis über Volumen führen, so sollte man strikte statt normale Ungleichungssysteme betrachten. Denn ein striktes Ungleichungssystem ist entweder unlösbar oder seine Lösungsmenge hat ein positives (relativ großes) Volumen.

Damit können wir die Ellipsoidmethode formulieren.

(4.26) Algorithmus (Ellipsoidmethode).
Eingabe: $A \in \mathbb{Q}^{(m,n)}$, $b \in \mathbb{Q}^m$.
Ausgabe: Ein Vektor $x \in \mathbb{Q}^n$ mit $Ax \leq b$ oder die Feststellung, dass $\{x \in \mathbb{R}^n \mid Ax < b\}$ leer ist.

1. **Initialisierung:** Setze

   $A_0 := R^2 I$, mit $R = \sqrt{n} \cdot 2^{2\langle A\rangle+\langle b\rangle-2n^2}$ (oder R durch andere Vorinformationen kleiner gewählt),

   $a_0 := 0$,

   $k := 0$,

   $N := 2n((3n+1)\langle A\rangle + (2n+1)\langle b\rangle - n^3)$.

   (Das Anfangsellipsoid ist $E_0 := E(A_0, a_0)$.)

2. **Abbruchkriterium:**

   (2.a) Gilt $k = N$, dann hat $Ax < b$ keine Lösung, STOP!

   (2.b) Gilt $Aa_k \leq b$, dann ist eine Lösung gefunden, STOP!

(2.c) Andernfalls sei c^T irgendeine Zeile von A derart, dass der Mittelpunkt a_k von E_k die entsprechende Ungleichung verletzt.

3. **Update:** Setze

(3.a) $a_{k+1} := a_k - \frac{1}{n+1}\frac{1}{\sqrt{c^T A_k c}} A_k c$,

(3.b) $A_{k+1} := \frac{n^2}{n^2-1}(A_k - \frac{2}{n+1}\frac{1}{c^T A_k c} A_k c c^T A_k^T)$ ($E_{k+1} := E(A_{k+1}, a_{k+1})$ ist das neue Ellipsoid), $k := k+1$.

4. Gehe zu 2. △

(4.27) Satz. *Die Ellipsoidmethode arbeitet korrekt.* △

Beweis. Gibt es ein $k \leq N$, so dass $Aa_k \leq b$, so ist offenbar ein Vektor aus $P(A,b)$ gefunden. Bricht die Ellipsoidmethode in Schritt (2.a) ab, so müssen wir zeigen, dass $\mathring{P} := \{x \mid Ax < b\}$ kein Element besitzt.

Angenommen $\mathring{P} \neq \emptyset$. Sei P' der Durchschnitt von $\mathring{P}$ mit E_0. Dann gilt nach Lemma (4.25), dass das Volumen von P' mindestens $2^{-(n+1)(\langle A\rangle+\langle b\rangle-n^2)}$ beträgt. Ist a_k nicht in $P(A,b)$, $0 \leq k < N$, so wird in (2.c) eine verletzte Ungleichung gefunden, sagen wir $c^T x \leq \gamma$. Wegen $\gamma < c^T a_k$ enthält das Halbellipsoid

$$E'_k := E_k \cap \{x \mid c^T x \leq c^T a_k\}$$

die Menge P'. Das durch die Formeln (3.a), (3.b) konstruierte neue Ellipsoid E_{k+1} ist nach Satz (4.20) das volumenmäßig kleinste Ellipsoid, das E'_k enthält. Wegen $P' \subseteq E'_k$ gilt natürlich $P' \subseteq E_{k+1}$. Daraus folgt

$$P' \subseteq E_k, \quad 0 \leq k \leq N.$$

Das Volumen des Anfangsellipsoids E_0 kann man wie folgt berechnen:

$$\mathrm{vol}(E_0) = \sqrt{\det(R^2 I)} \cdot V_n = R^n V_n,$$

wobei V_n das Volumen der Einheitskugel im $\mathbb{R}^n$ ist. Wir machen nun eine sehr grobe Abschätzung. Die Einheitskugel ist im Würfel $W = \{x \in \mathbb{R}^n \mid |x_i| \leq 1, i = 1, \ldots, n\}$ enthalten. Das Volumen von W ist offensichtlich 2^n. Daraus folgt

$$\mathrm{vol}(E_0) \;<\; R^n 2^n \;=\; 2^{n(2\langle A\rangle+\langle b\rangle-2n^2+\log\sqrt{n}+1)} \;<\; 2^{n(2\langle A\rangle+\langle b\rangle-n^2)}.$$

In jedem Schritt der Ellipsoidmethode schrumpft nach (4.24) das Volumen um mindestens den Faktor $e^{-\frac{1}{2n}}$. Aus der Abschätzung von $\mathrm{vol}(E_0)$ und der in 1. angegebenen Formel für N erhalten wir somit

$$\mathrm{vol}(E_N) \;\leq\; e^{-\frac{N}{2n}}\,\mathrm{vol}(E_0) \;<\; 2^{-(n+1)(\langle A\rangle+\langle b\rangle-n^2)}.$$

Also gilt $\mathrm{vol}(E_N) < \mathrm{vol}(P')$ und $P' \subseteq E_N$. Dies ist ein Widerspruch. Hieraus folgt, dass P' und somit $\{x \mid Ax < b\}$ leer sind, wenn die Ellipsoidmethode in (2.a) abbricht. □

Aus (4.25)(b) folgt nun unmittelbar

(4.28) Korollar. *Ist $P = P(A,b) \subseteq \mathbb{R}^n$ ein Polyeder, von dem wir wissen, dass es entweder volldimensional oder leer ist, dann findet die Ellipsoidmethode entweder einen Punkt in P oder beweist, dass P leer ist.* △

Nun kann man natürlich einem durch ein Ungleichungssystem gegebenen Polyeder nicht unmittelbar ansehen, ob es volldimensional ist oder nicht. Ferner können gerade diejenigen Polyeder, die durch Transformation linearer Programme entstehen (siehe (4.4), hier gilt immer $c^T x = b^T y$), nicht volldimensional sein, so dass die Ellipsoidmethode zu keiner befriedigenden Antwort führt. Diesen Defekt kann man jedoch durch die folgende Beobachtung reparieren.

(4.29) Satz. *Seien $A \in \mathbb{Q}^{(m,n)}$ und $b \in \mathbb{Q}^m$, dann hat das Ungleichungssystem*

$$Ax \leq b$$

genau dann eine Lösung, wenn das strikte Ungleichungssystem

$$Ax < b + 2^{-2\langle A\rangle - \langle b\rangle}\mathbb{1}$$

eine Lösung hat. Ferner kann man aus einer Lösung des strikten Ungleichungssystems in polynomialer Zeit eine Lösung von $Ax \leq b$ konstruieren. △

Damit ist die Beschreibung der Ellipsoidmethode (bis auf die Abschätzung der Rechenzeit) vollständig. Wollen wir entscheiden, ob ein Polyeder $P(A,b)$ einen Punkt enthält, können wir die Methode (4.26) auf das Ungleichungssystem $Ax \leq b$ anwenden. Finden wir einen Punkt in $P(A,b)$, dann sind wir fertig. Andernfalls wissen wir, dass $P(A,b)$ nicht volldimensional ist. In diesem Falle können wir auf Satz (4.29) zurückgreifen und starten die Ellipsoidmethode neu, und zwar z. B. mit dem ganzzahligen Ungleichungssystem

$$2^{2\langle A\rangle + \langle b\rangle} Ax \leq 2^{2\langle A\rangle + \langle b\rangle} b + \mathbb{1}. \tag{4.30}$$

Entscheidet die Ellipsoidmethode, dass das zu (4.30) gehörige strikte Ungleichungssystem keine Lösung hat (Abbruch in Schritt (2.a)), so können wir aus (4.29) folgern, dass $P(A,b)$ leer ist. Andernfalls findet die Ellipsoidmethode einen Vektor x', der (4.30) erfüllt. Gilt $x' \in P(A,b)$, haben wir das gewünschte gefunden, falls nicht, kann man (mit einfachen Methoden der linearen Algebra) aus x' einen Punkt $x \in P(A,b)$ konstruieren, siehe (4.29).

Zur Lösung linearer Programme kann man die in Abschnitt 4.1 besprochenen Reduktionen benutzen. Entweder man fasst das lineare Programm (4.2) und das dazu duale (4.3) zu (4.4) zusammen und sucht wie oben angegeben im nicht volldimensionalen Polyeder (4.4) einen Punkt, oder man wendet $\overline{N}$-mal, siehe (4.14), die Ellipsoidmethode für niederdimensionale Polyeder des Typs P_s aus (4.12)(2) im Rahmen eines binären Suchverfahrens (4.12) an.

In jedem Falle ist das Gesamtverfahren polynomial, wenn die Ellipsoidmethode (4.26) polynomial ist.

4.3 Laufzeit der Ellipsoidmethode

Wir wollen nun die Laufzeit der Ellipsoidmethode untersuchen und auf einige bisher verschwiegene Probleme bei der Ausführung von (4.26) aufmerksam machen. Offenbar ist die maximale Iterationszahl

$$N = 2n((3n+1)\langle A\rangle + (2n+1)\langle b\rangle - n^3)$$

polynomial in der Kodierungslänge von A und b. Also ist das Verfahren (4.26) genau dann polynomial, wenn jede Iteration in polynomialer Zeit ausgeführt werden kann.

Bei der Initialisierung 1 besteht kein Problem. Test (2.a) ist trivial, und die Schritte (2.b) und (2.c) führen wir dadurch aus, dass wir das Zentrum a_k in die Ungleichungen einsetzen und überprüfen, ob die Ungleichungen erfüllt sind oder nicht. Die Anzahl der hierzu benötigten elementaren Rechenschritte ist linear in $\langle A\rangle$ und $\langle b\rangle$. Sie ist also polynomial, wenn die Kodierungslänge des Vektors a_{k+1} polynomial ist.

An dieser Stelle beginnen die Schwierigkeiten. In der Update-Formel (3.a) muss eine Wurzel berechnet werden. I. A. werden also hier irrationale Zahlen auftreten, die natürlich nicht exakt berechnet werden können. Die (möglicherweise) irrationale Zahl $\sqrt{c^T A_k c}$ muss daher zu einer rationalen Zahl gerundet werden. Dadurch wird geometrisch bewirkt, dass der Mittelpunkt des Ellipsoids E_{k+1} ein wenig verschoben wird. Mit Sicherheit enthält das so verschobene Ellipsoid nicht mehr die Menge E'_k (siehe Satz (4.20)) und möglicherweise ist auch $P(A,b)$ nicht mehr in diesem Ellipsoid enthalten. Also bricht unser gesamter Beweis der Korrektheit des Verfahrens zusammen.

Ferner wird beim Update (3.b) durch möglicherweise große Zahlen geteilt, und es ist nicht a priori klar, dass die Kodierungslänge der Elemente von A_{k+1} bei wiederholter Anwendung von (3.b) polynomial in $\langle A\rangle + \langle b\rangle$ bleibt. Also müssen auch die Einträge in A_{k+1} gerundet werden. Dies kann zu folgenden Problemen führen. Die gerundete Matrix, sagen wir A^*_{k+1}, ist nicht mehr positiv definit und das Verfahren wird sinnlos. Oder A^*_{k+1} bleibt positiv definit, aber durch die Rundung hat sich die Form des zugehörigen Ellipsoids, sagen wir E^*_{k+1} so geändert, dass E^*_{k+1} das Polyeder $P(A,b)$ nicht mehr enthält.

Alle diese Klippen kann man mit einem einfachen Trick umschiffen, dessen Korrektheitsbeweis allerdings recht aufwendig ist. Die geometrische Idee hinter diesem Trick ist die folgende. Man nehme die binäre Darstellung der Komponenten des in (3.a) berechneten Vektors und der in (3.b) berechneten Matrix und runde nach p Stellen hinter dem Binärkomma. Dadurch ändert man die Lage des Mittelpunkts und die Form des Ellipsoids ein wenig. Nun bläst man das Ellipsoid ein bisschen auf, d. h. man multipliziert A_{k+1} mit einem Faktor $\zeta > 1$ und zwar so, dass die Menge E'_k beweisbar in dem aufgeblasenen Ellipsoid enthalten ist. Durch die Vergrößerung des Ellipsoids wird natürlich die in (4.24) bestimmte Schrumpfungsrate verschlechtert, was bedeutet, dass man insgesamt mehr Iterationen, sagen wir N', durchführen muss. Daraus folgt, dass der Rundungsparameter p und der Aufblasparameter ζ aufeinander so abgestimmt sein müssen, dass alle gerundeten und mit ζ multiplizierten Matrizen A_k, $1 \le k \le N'$ und alle gerundeten Mittelpunkte a_k, $1 \le k \le N'$ polynomial in $\langle A\rangle + \langle b\rangle$ berechnet werden können, so dass alle A_k positiv definit sind, $P \cap S(0,R)$ in allen Ellipsoiden E_k enthalten ist und die Iterationszahl N' ebenfalls polynomial in $\langle A\rangle + \langle b\rangle$ ist. Dies kann in der Tat durch

Anwendung von Abschätzungstechniken aus der Analysis und linearen Algebra realisiert werden, siehe Grötschel et al. (1988).

Die Ellipsoidmethode (mit Rundungsmodifikation) ist also ein polynomialer Algorithmus, der entscheidet, ob ein Polyeder leer ist oder nicht. Mit Hilfe der im Vorhergehenden beschriebenen Reduktion kann sie dazu benutzt werden, lineare Programme in polynomialer Zeit zu lösen.

Wie sich der Leser denken kann, gibt es eine ganze Reihe von Varianten der Ellipsoidmethode, die dazu dienen sollen, schnellere Konvergenz zu erzwingen. Derartige Varianten sind z. B. in Bland et al. (1981), Grötschel et al. (1988) und Akgül (1984) beschrieben. Aus Platz- und Zeitgründen können wir hier nicht weiter darauf eingehen.

4.4 Ein Beispiel

Wir wollen hier den Ablauf der Ellipsoidmethode anhand eines Beispiels im $\mathbb{R}^2$ geometrisch veranschaulichen. Wir starten mit dem folgenden Polyeder $P(A, b) \subseteq \mathbb{R}^2$ definiert durch

$$\begin{aligned} -x_1 - x_2 &\leq -2 \\ 3x_1 &\leq 4 \\ -2x_1 + 2x_2 &\leq 3 \end{aligned}$$

Dieses Polyeder P ist in der Kugel (Kreis) um den Nullpunkt mit Radius 7 enthalten. Diese Kugel soll unser Anfangsellipsoid $E_0 = E(A_0, 0)$ sein. Wir führen mit dieser Initialisierung die Schritte 2 und 3 des Ellipsoidverfahrens (4.26) durch. Wir rechnen natürlich nicht mit der eigentlich erforderlichen Genauigkeit sondern benutzen die vorhandene Maschinenpräzision. In unserem Fall ist das Programm in PASCAL und die Rechnungen werden in `REAL`-Arithmetik durchgeführt, d. h. wir benutzen eine Mantisse von 3 byte und einen Exponenten von 1 byte zur Zahlendarstellung. Das Verfahren führt insgesamt 7 Iterationen durch und endet mit einem zulässigen Punkt. In den nachfolgenden Abbildungen 4.3 bis 4.10 sind (im Maßstab 1 : 2) jeweils die Ellipsoide E_k und E_{k+1} mit ihren Mittelpunkten a_k und a_{k+1}, $k = 0, \ldots, 6$ aufgezeichnet. Man sieht also, wie sich die Form und Lage eines Ellipsoids in einem Iterationsschritt ändert. Das Startellipsoid ist gegeben durch

$$a_0^T = (0, 0), \qquad A = \begin{pmatrix} 49 & 0 \\ 0 & 49 \end{pmatrix}.$$

Neben jeder Abbildung sind das neue Zentrum a_{k+1} und die neue Matrix A_{k+1} mit $E_{k+1} = E(A_{k+1}, a_{k+1})$ angegeben. Jede Abbildung enthält die Begrenzungsgeraden des Polytops P, so dass man auf einfache Weise sehen kann, welche Ungleichungen durch den neuen Mittelpunkt verletzt sind.

Literaturverzeichnis

M. Akgül. *Topics in relaxation and ellipsoidal methods*, Band 97 von *Research Notes in Mathematics*. Pitman Advanced Publishing Program. VII, Boston, 1984.

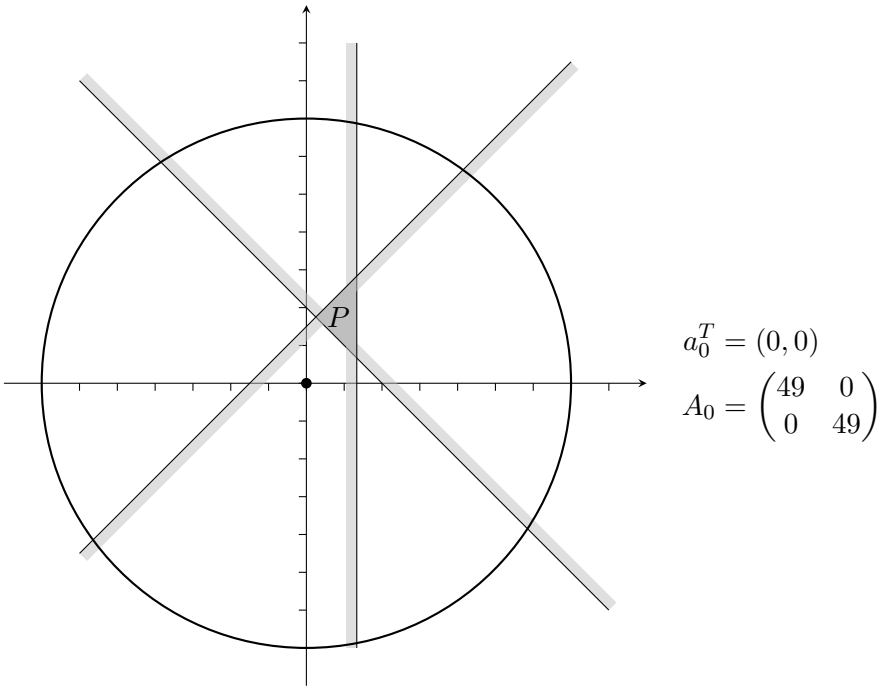


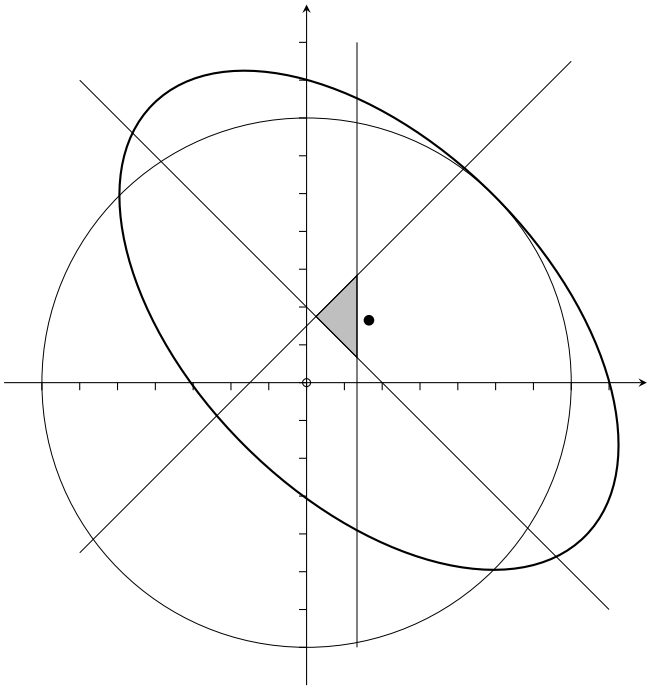
Abbildung 4.3: Polytop P und Startellipsoid E_0

R. G. Bland, D. Goldfarb, und M. J. Todd. The ellipsoid method: A survey. *Operations Research*, 29:1039–1091, 1981.

K.-H. Borgwardt. The average number of pivot steps required by the simplex-method is polynomial. *Zeitschrift für Operations Research, Serie A*, 26:157–177, 1982.

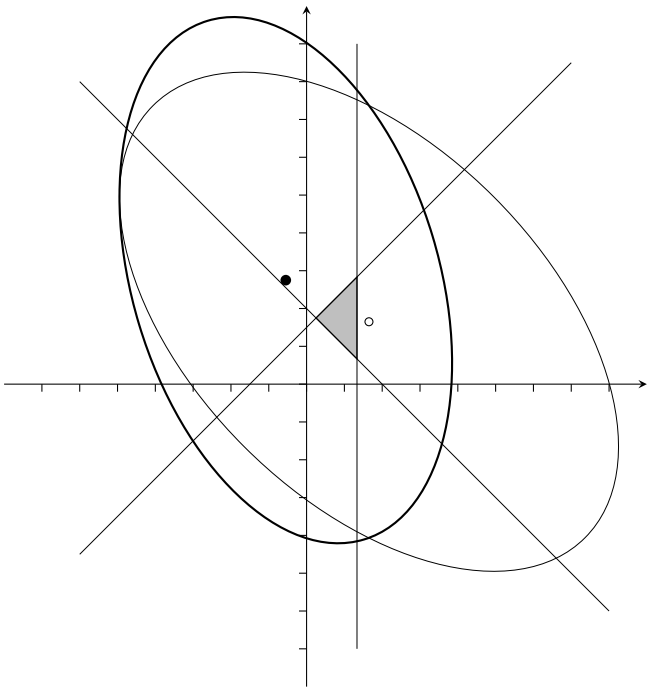
M. Grötschel, L. Lovász, und A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1988.

L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademia Nauk SSSR*, 244:1093–1096, 1979. Englische Übersetzung in Soviet Mathematics Doklady, 20:191–194, 1979.



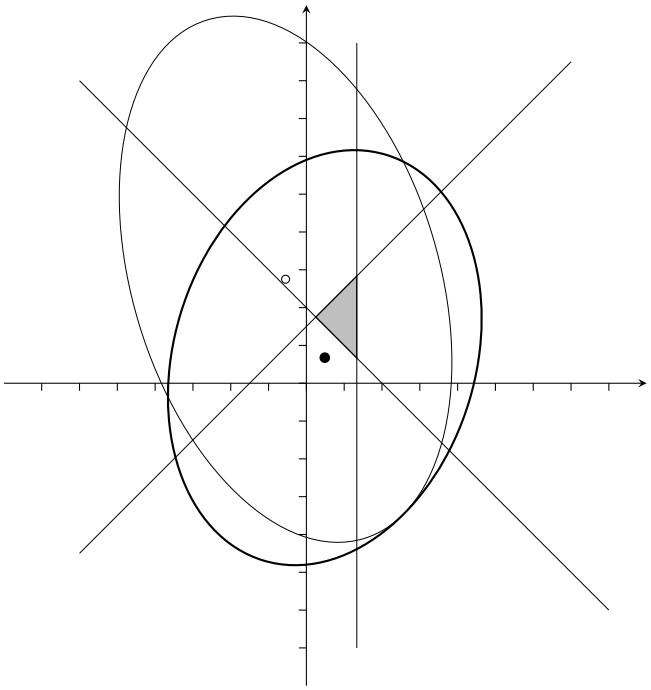
$$a_1^T = (1.6499, 1.6499)$$
$$A_1 = \begin{pmatrix} 43.5555 & -21.7777 \\ -21.7777 & 43.5555 \end{pmatrix}$$

Abbildung 4.4: Iteration 1 (E_1 und sein Mittelpunkt sind fett gezeichnet)



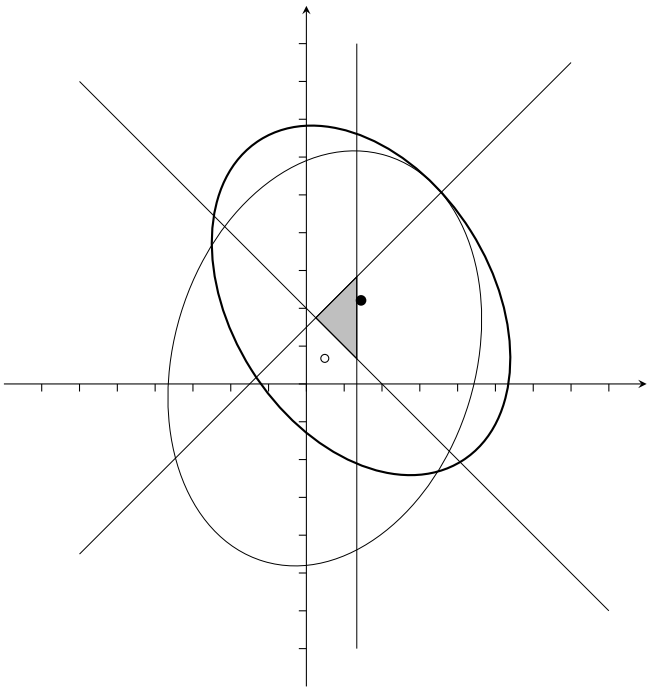
$$a_1^T = (-0.5499, 2.7499)$$
$$A_1 = \begin{pmatrix} 19.3580 & -9.6790 \\ -9.6790 & 48.3951 \end{pmatrix}$$

Abbildung 4.5: Iteration 2



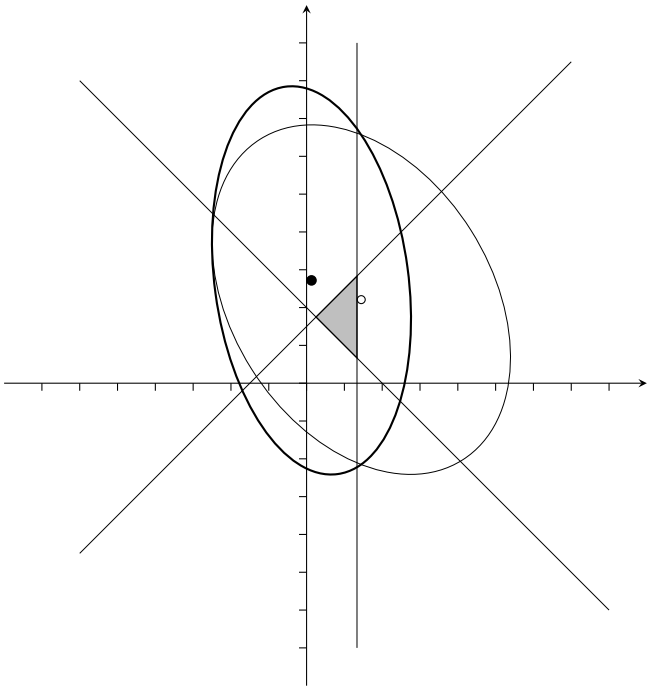
$$a_1^T = (0.4871, 0.6758)$$
$$A_1 = \begin{pmatrix} 17.2071 & 4.3018 \\ 4.3018 & 30.1125 \end{pmatrix}$$

Abbildung 4.6: Iteration 3



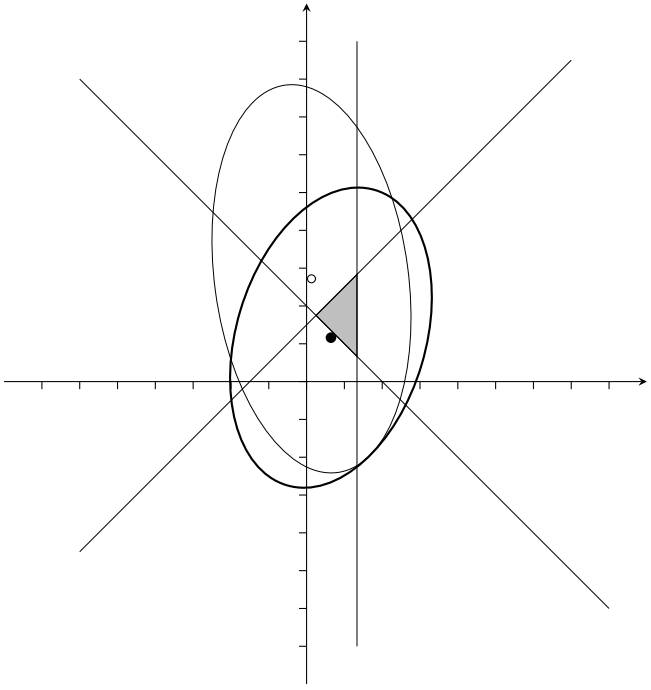
$$a_1^T = (1.4458, 2.2098)$$
$$A_1 = \begin{pmatrix} 15.5894 & -6.0299 \\ -6.0299 & 21.3510 \end{pmatrix}$$

Abbildung 4.7: Iteration 4



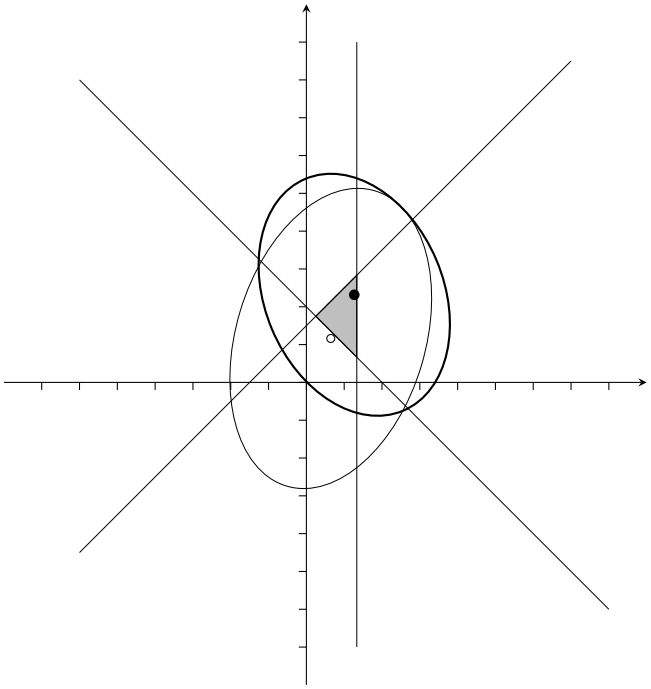
$$a_1^T = (0.1297, 2.7188)$$
$$A_1 = \begin{pmatrix} 6.9286 & -2.6799 \\ -2.6799 & 26.3603 \end{pmatrix}$$

Abbildung 4.8: Iteration 5



$$a_1^T = (0.6449, 1.1618)$$
$$A_1 = \begin{pmatrix} 7.1148 & 2.8443 \\ 2.8443 & 15.7511 \end{pmatrix}$$

Abbildung 4.9: Iteration 6



$$a_1^T = (1.2661, 2.3170)$$

$$A_1 = \begin{pmatrix} 6.3988 & -1.9726 \\ -1.9726 & 10.2372 \end{pmatrix}$$

Abbildung 4.10: Iteration 7 (mit zulässigem Punkt a_7)

5 Innere-Punkte-Verfahren

Die Ellipsoidmethode war ein theoretischer Durchbruch aber kein praktischer Erfolg. Fünf Jahre nach Khachiyans Publikation kündigte sich dann eine weitere Sensation an. Eine Idee, die bereits in den 1950er und 1960er Jahren untersucht, aber nie zu einem ernsthaften Konkurrenten des Simplexverfahrens entwickelt wurde, sollte nun sowohl theoretisch als auch praktisch zur Lösung von linearen Optimierungsaufgaben geeignet sein. Der Vorschlag ist, zunächst einen Punkt im Inneren des Polyeders zu konstruieren und dann durch das Innere des Polyeders entlang eines Pfades (den man auf verschiedene Weisen definieren kann) auf eine Optimallösung zuzusteuern. Narendra Karmarkar löste durch seinen Artikel und seine Vorträge im Jahr 1984 eine neue Welle von Algorithmenentwicklungen aus, die dann tatsächlich zu Verfahren geführt hat, die in vielen Fällen, insbesondere bei sehr großen LPs, schneller als der Simplexalgorithmus arbeiten. Die Entstehungsgeschichte der inzwischen Innere-Punkte- oder Barriere-Verfahren genannten Algorithmen ist sehr spannend und berührt auch das Thema „Patentierbarkeit von Algorithmen“. Wir verweisen hierzu auf den Artikel *Who Invented the Interior-Point Method?*, siehe Shanno (2012).

In diesem Kapitel stellen wir zunächst ausführlich die Idee des Karmarkar-Algorithmus dar, danach erläutern wir primale sowie primal-duale Pfadverfolgungsalgorithmen. Numerische Details (die gute Kenntnisse der nichtlinearen Optimierung voraussetzen) können hier – auch aus Zeitgründen – nicht dargestellt werden.

5.1 Der Karmarkar-Algorithmus

Im Jahre 1984 hat N. Karmarkar (AT&T Bell Laboratories) einen neuen Algorithmus zur Lösung linearer Programme entwickelt und in Karmarkar (1984) veröffentlicht. Im Herbst 1984 hat Karmarkar auf Vorträgen mitgeteilt, dass sein Verfahren nicht nur theoretisch polynomial ist (wie auch die in Kapitel 4 beschriebene Ellipsoidmethode), sondern in der Praxis (Implementation bei AT&T Bell Laboratories) erheblich rascher als das Simplexverfahren arbeitet. Karmarkar behauptete, dass die Implementation seines Algorithmus etwa 50-mal schneller ist als MPSX, ein von der Firma IBM angebotenes und auf dem Simplexalgorithmus basierendes Softwarepaket zur Lösung linearer Programme, das zur damaligen Zeit als eines der besten LP-Pakete galt.

Da bei großen Industriefirmen sehr große LPs täglich in nicht unerheblicher Zahl gelöst werden, war die Ankündigung Karmarkars auf sehr großes Interesse bei Praktikern gestoßen. Ja, sein Verfahren hat sogar Aufsehen in der Presse verursacht. Zum Beispiel haben New York Times, Science, Time, Der Spiegel (falsch – wie bei Artikeln über Mathematik üblich), Die Zeit, Süddeutsche Zeitung (sehr ordentlich) über Karmarkars Verfahren berichtet, wie immer natürlich unter dem Hauptaspekt, dass dadurch Millionen von Dollars

gespart werden können. Im Nachfolgenden wollen wir die Grundidee des Algorithmus von Karmarkar darstellen.

Reduktionen

Wie der Simplexalgorithmus und die Ellipsoidmethode, so ist auch der Karmarkar-Algorithmus auf einen speziellen Problemtyp zugeschneidert und nicht auf allgemeine lineare Programme anwendbar. Wie üblich müssen wir daher zunächst zeigen, dass ein allgemeines LP so transformiert werden kann, dass es mit Karmarkars Algorithmus gelöst werden kann. Die Grundversion des Karmarkar-Algorithmus (und nur die wollen wir hier beschreiben) ist ein Verfahren zur Entscheidung, ob ein Polyeder einer recht speziellen Form einen Punkt enthält oder nicht. Das Problem, das Karmarkar betrachtet, ist das folgende.

(5.1) Problem. Gegeben seien eine Matrix $A \in \mathbb{Q}^{(m,n)}$ und ein Vektor $c \in \mathbb{Q}^n$. Entscheide, ob das System

$$Ax = 0, \quad \mathbb{1}^T x = 1, \quad x \geq 0, \quad c^T x \leq 0$$

eine Lösung hat, und falls das so ist, finde eine. △

Um den Karmarkar-Algorithmus starten zu können, ist die Kenntnis eines zulässigen Startvektors im relativ Inneren von $\{x \mid Ax = 0, \mathbb{1}^T x = 1, x \geq 0\}$ notwendig. Wir wollen sogar noch mehr fordern, und zwar soll gelten

$$\overline{x} := \frac{1}{n}\mathbb{1} \quad \text{erfüllt} \quad A\overline{x} = 0. \tag{5.2}$$

Trivialerweise erfüllt $\overline{x}$ die Bedingungen $\mathbb{1}^T\overline{x} = 1$, $\overline{x} \geq 0$. Gilt $c^T\overline{x} \leq 0$, ist man natürlich fertig. Die eigentliche Aufgabe besteht also darin, aus $\overline{x}$ einen Punkt zu konstruieren, der alle Bedingungen von (5.1) erfüllt.

Wir werden nun, wie bei der Darstellung der Ellipsoidmethode, ein allgemeines LP in ein (bzw. mehrere) Probleme des Typs (5.1) umformen.

Wir wissen bereits, dass jedes LP in ein LP in Standardform (siehe ADM I, Definition (9.1)) transformiert werden kann. Gegeben sei also das folgende Problem

$$\begin{aligned} \max \quad & w^T x \\ & Bx = b \\ & x \geq 0 \end{aligned} \tag{5.3}$$

mit $w \in \mathbb{Q}^n$, $B \in \mathbb{Q}^{(m,n)}$, $b \in \mathbb{Q}^m$. Wie in Abschnitt 4.1 gezeigt gibt es zwei prinzipielle Reduktionsmöglichkeiten. Man fasst (5.3) und das dazu duale Problem wie in (4.4) zusammen, oder man führt die in (4.12) beschriebene Binärsuche durch. Wir stellen hier kurz die Reduktion über Binärsuche dar. Daraus ergibt sich auch, wie man die Kombination des primalen mit dem dualen Problem behandeln kann.

Genau wie in (4.12) angegeben (und mit den in den davor angegebenen Sätzen vorgelegten Begründungen) führen wir eine binäre Suche durch über der Menge

$$S := \left\{ \frac{p}{q} \in Q \mid |p| \leq n \cdot 2^{\langle B\rangle+\langle b\rangle+2\langle w\rangle-n^2-n}, 1 \leq q \leq 2^{\langle B\rangle+\langle w\rangle-n^2-n} \right\}$$

der möglichen optimalen Zielfunktionswerte (falls (5.3) eine endliche Optimallösung hat). Zur Bestimmung einer optimalen Lösung von (5.3) sind nach (4.14) höchstens

$$\overline{N} := 2\langle B\rangle + \langle b\rangle + 3\langle w\rangle - 2n^2 - 2n + \log_2 n + 2$$

Aufrufe eines Algorithmus nötig, der für ein $s \in S$ entscheidet, ob

$$\begin{aligned} w^T x &\leq s \\ Bx &= b \\ x &\geq 0 \end{aligned} \tag{5.4}$$

eine Lösung hat. Genau dann, wenn wir zeigen können, dass (5.4) in polynomialer Zeit gelöst werden kann, haben wir also ein polynomiales Verfahren zur Lösung von (5.3). Wir wissen aus Satz (4.7), dass wir alle Variablen durch $2^{2\langle B\rangle+\langle b\rangle-2n^2}$ beschränken können. Führen wir für die Ungleichung in (5.4) eine Schlupfvariable x_{n+1} ein, so gilt $0 \leq x_{n+1} \leq \max S = n \cdot 2^{\langle B\rangle+\langle b\rangle+2\langle w\rangle-n^2-n}$. Setzen wir also

$$M := n(2^{2\langle B\rangle+\langle b\rangle-2n^2} + 2^{\langle B\rangle+\langle b\rangle+2\langle w\rangle-n^2-n}),$$

so ist (5.4) genau dann lösbar, wenn

$$\begin{aligned} \begin{pmatrix} w^T & 1 \\ B & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{n+1} \end{pmatrix} &= \begin{pmatrix} s \\ b \end{pmatrix} \\ \sum_{i=1}^{n+1} x_i &\leq M \\ x_i &\geq 0, \quad i = 1, \ldots, n+1 \end{aligned} \tag{5.5}$$

lösbar ist. Führen wir eine weitere Schlupfvariable x_{n+2} ein (d. h. es gilt nun $x \in \mathbb{R}^{n+2}$) und skalieren wir die Variablen um, so ist (5.5) genau dann lösbar, wenn

$$\begin{aligned} Cx := \quad \begin{pmatrix} w^T & 1 & 0 \\ B & 0 & 0 \end{pmatrix} x = \frac{1}{M} \begin{pmatrix} s \\ b \end{pmatrix} \quad =: \begin{pmatrix} \overline{c}_0 \\ \vdots \\ \overline{c}_m \end{pmatrix} \\ \mathbb{1}^T x = 1 \\ x_i \geq 0, \quad i = 1, \ldots, n+2 \end{aligned} \tag{5.6}$$

lösbar ist. Von der i-ten Zeile der Matrix C ($i = 0, \ldots, m$) ziehen wir nun das $\overline{c}_i$-fache der letzten Gleichung $\mathbb{1}^T x = 1$ ab. Dadurch machen wir die ersten $m+1$ Gleichungen

homogen und erreichen, dass aus dem Ungleichungs- und Gleichungssystem (5.6) ein äquivalentes System der folgenden Form wird:

$$\begin{aligned} Dx &= 0 \\ \mathbb{1}^T x &= 1 \\ x &\geq 0 \end{aligned} \tag{5.7}$$

Um die Voraussetzung (5.2) herzustellen, führen wir eine weitere Schlupfvariable x_{n+3} wie folgt ein. Wir betrachten

$$\begin{aligned} Dx - D\mathbb{1}x_{n+3} &= 0 \\ \mathbb{1}^T x + x_{n+3} &= 1 \\ x_i \geq 0, \quad i &= 1, \ldots, n+3. \end{aligned} \tag{5.8}$$

Offenbar hat (5.7) genau dann eine Lösung, wenn (5.8) eine Lösung mit $x_{n+3} = 0$ hat. Setzen wir

$$A := (D, -D\mathbb{1}) \in \mathbb{Q}^{(m+1,n+3)}, \qquad c := (0, \ldots, 0, 1)^T \in \mathbb{Q}^{n+3}$$

und betrachten wir x als Vektor im $\mathbb{R}^{n+3}$, so hat also (5.7) genau dann eine Lösung, wenn

$$Ax = 0, \quad \mathbb{1}^T x = 1, \quad x \geq 0, \quad c^T x \leq 0 \tag{5.9}$$

eine Lösung hat. Ferner erfüllt nach Konstruktion der Vektor

$$\overline{x}^T := \left(\frac{1}{n+3}, \ldots, \frac{1}{n+3} \right) \in \mathbb{Q}^{n+3}$$

bezüglich des Systems (5.9) die Forderung (5.2).

Folglich kann jedes lineare Program durch polynomial viele Aufrufe eines Algorithmus zur Lösung von (5.1) (mit Zusatzvoraussetzung (5.2)) gelöst werden.

Die Grundversion des Karmarkar-Algorithmus

Wir wollen nun das Karmarkar-Verfahren zur Lösung von (5.1) unter der Zusatzvoraussetzung (5.2) beschreiben. Wie wir am Ende des letzten Abschnittes gesehen haben, können wir jedes LP in eine Folge von Problemen der Form (5.1) transformieren, und wir können sogar das Zentrum des Simplex $\{x \in \mathbb{R}^n \mid \mathbb{1}^T x = 1, x \geq 0\}$ als Startpunkt wählen. Im weiteren betrachten wir also das folgende

(5.10) Problem. Gegeben seien eine Matrix $A \in \mathbb{Q}^{(m,n)}$ und ein Vektor $c \in \mathbb{Q}^n$. Es seien

$$\begin{aligned} E &:= \{x \in \mathbb{R}^n \mid \mathbb{1}^T x = 1\} && \text{(Hyperebene)} \\ \Sigma &:= E \cap \mathbb{R}^n_+ && \text{(Simplex)} \\ \Omega &:= \{x \in \mathbb{R}^n \mid Ax = 0\} && \text{(linearer Raum).} \end{aligned}$$

Ferner wird vorausgesetzt, dass der Vektor $\overline{x} := \frac{1}{n}\mathbb{1}$ in $\Sigma \cap \Omega$ enthalten ist. Gesucht ist ein Vektor $x \in \mathbb{Q}^n$ mit

$$x \in P := \Sigma \cap \Omega, \quad c^T x \leq 0. \qquad \triangle$$

Problem (5.10) kann sicherlich dadurch gelöst werden, daß eine Optimallösung des folgenden Programms bestimmt wird.

$$\begin{array}{rl} \min & c^T x \\ & Ax = 0 \\ & \mathbb{1}^T x = 1 \\ & x \geq 0 \end{array} \quad \text{bzw.} \quad \begin{array}{l} \min c^T x \\ x \in \Omega \cap E \cap \mathbb{R}^n_+ \end{array} \tag{5.11}$$

Offenbar ist der optimale Zielfunktionswert von (5.11) genau dann größer als Null, wenn (5.10) keine Lösung hat. Unser Ziel wird nun sein (5.11) statt (5.10) zu lösen, wobei wir voraussetzen, dass $\frac{1}{n}\mathbb{1}$ für (5.11) zulässig ist.

Es erscheint natürlich reichlich umständlich, ein lineares Programm, sagen wir der Form (5.3), durch Binärsuche auf eine Folge von Zulässigkeitsproblemen (5.10) zu reduzieren, die dann wieder durch spezielle lineare Programme der Form (5.11) gelöst werden. Diesen Umweg kann man durch die sogenannte „Sliding Objective Function Technique“ begradigen. Die dabei notwendigen zusätzlichen Überlegungen tragen jedoch nicht zu einem besseren Verständnis der Grundversion des Verfahrens bei, um die es hier geht. Ziel dieses Kapitels ist nicht die Darstellung einer besonders effizienten Version des Karmarkar-Algorithmus sondern dessen prinzipielle Idee.

Geometrische Beschreibung eines Iterationsschrittes

Zur Lösung von (5.11) hat Karmarkar ein Verfahren entworfen, das wie fast alle Optimierungsverfahren (insbesondere die Methoden der nichtlinearen Optimierung) auf der folgenden simplen Idee beruht. Angenommen man befinde sich an einem zulässigen Punkt, sagen wir x^k, dann sucht man eine Richtung (also einen Vektor $d \in \mathbb{R}^n$), bezüglich der die Zielfunktion verbessert werden kann. Daraufhin bestimmt man eine Schrittlänge ρ, so dass man von x^k zum nächsten Punkt $x^{k+1} := x^k + \rho d$ gelangt. Der nächste Punkt x^{k+1} soll natürlich auch zulässig sein und einen „wesentlich“ besseren Zielfunktionswert haben.

Die Essenz eines jeden solchen Verfahrens steckt natürlich in der Wahl der Richtung und der Schrittlänge. Bei derartigen Verfahren tritt häufig die folgende Situation ein. Man ist in der Lage, eine sehr gute Richtung zu bestimmen (d. h. die Zielfunktion wird in Richtung d stark verbessert), aber man kann in Richtung d nur einen sehr kleinen Schritt ausführen, wenn man die zulässige Menge nicht verlassen will. Trotz guter Richtung kommt man also im Bezug auf eine tatsächliche Verbesserung kaum vorwärts und erhält u. U. global schlechtes Konvergenzverhalten. Man muss sich also bemühen, einen guten Kompromiss zwischen „Qualität der Richtung“ und „mögliche Schrittlänge“ zu finden, um insgesamt gute Fortschritte zu machen.

Ist man – wie im vorliegenden Fall – im relativen Inneren der Menge P, aber nah am Rand und geht man z. B. in Richtung des Normalenvektors der Zielfunktion, so kann man sehr schnell an den Rand von P gelangen, ohne wirklich weiter gekommen zu sein, siehe Abbildung 5.1. Karmarkars Idee zur Lösung bzw. Umgehung dieser Schwierigkeit

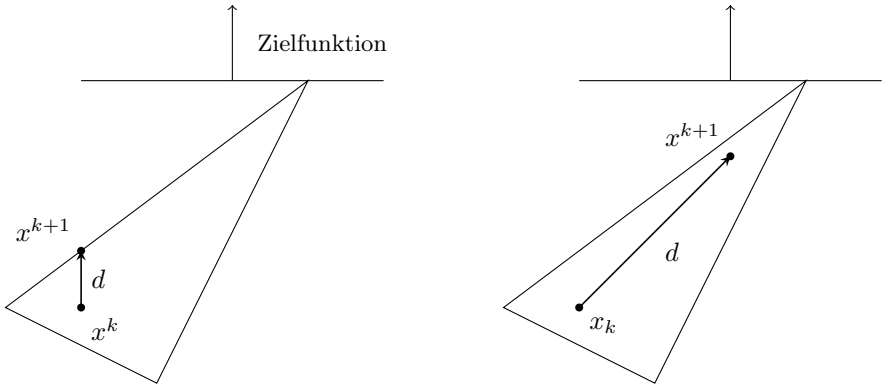


Abbildung 5.1: Gute Richtung, geringer Fortschritt (links), schlechtere Richtung, aber großer Fortschritt (rechts)

ist die folgende. Er führt eine projektive Transformation aus, die den Simplex Σ auf sich selbst, den affinen Teilraum Ω auf einen anderen affinen Teilraum Ω' abbildet und den relativ inneren Punkt x^k auf das Zentrum $\frac{1}{n}\mathbb{1}$ von Σ wirft. P wird dabei auf ein neues Polyeder P_k abgebildet. Offenbar kann man von $\frac{1}{n}\mathbb{1}$ aus recht große Schritte in alle zulässigen Richtungen machen, ohne sofort den zulässigen Bereich P_k zu verlassen. Man bestimmt so durch Festlegung einer Richtung und einer Schrittlänge von $\frac{1}{n}\mathbb{1}$ ausgehend einen Punkt $y^{k+1} \in P_k$ und transformiert diesen zurück, um den nächsten (zulässigen) Iterationspunkt $x^{k+1} \in P$ zu erhalten.

Zur Bestimmung der Richtung macht Karmarkar folgende Überlegung. Ideal wäre es natürlich, direkt das Optimum des transformierten Problems zu bestimmen. Aber die lineare Zielfunktion des ursprünglichen Problems wird durch die projektive Transformation in eine nichtlineare Abbildung übergeführt, so dass dies nicht so einfach zu bewerkstelligen ist. Dennoch kann man diese transformierte Zielfunktion auf natürliche Weise linearisieren. Mit $\overline{c}^T x$ sei die neue (linearisierte) Zielfunktion bezeichnet. Man hat nun ein neues Problem: Es soll eine lineare Zielfunktion über dem Durchschnitt eines Simplex mit einem affinen Raum optimiert werden, wir erhalten

$$\begin{aligned} &\min \overline{c}^T x \\ &\quad x \in \Omega' \cap E \cap \mathbb{R}^n_+ \end{aligned} \tag{5.12}$$

Dieses Problem ist offenbar wiederum vom Typ unseres Ausgangsproblems (5.11). Aber diese neue Aufgabe ist ja nur ein Hilfsproblem! Möglicherweise ist daher eine gute Approximation der Optimallösung von (5.12) ausreichend für das, was wir bezwecken. Durch die Lösung einer Vereinfachung dieses Problems (5.12) könnten u. U. ein Richtungsvektor und eine Schrittlänge gefunden werden, die von hinreichend guter Qualität bezüglich

globaler Konvergenz des Verfahrens sind. Die Vereinfachung, die wir betrachten wollen, geschieht dadurch, dass die bei der Durchschnittsbildung beteiligte Menge $\mathbb{R}^n_+$ durch die größte Kugel, sagen wir K, mit Zentrum $\frac{1}{n}\mathbb{1}$, die im Simplex Σ enthalten ist, ersetzt wird. Statt (5.12) wird also die Aufgabe

$$\begin{aligned} &\min \overline{c}^T x \\ &\quad x \in \Omega' \cap E \cap K \end{aligned} \tag{5.13}$$

betrachtet. Man optimiere eine lineare Zielfunktion über dem Durchschnitt einer Kugel K mit einem affinen Raum $\Omega' \cap E$. Dieses Problem ist ohne Einschaltung eines Algorithmus durch eine explizite Formel lösbar. Durch die Optimallösung von (5.13) werden die gesuchte Richtung und die Schrittlänge explizit geliefert.

Eine Modifikation ist jedoch noch nötig. Das Optimum über $\Omega' \cap E \cap K$ ist i. A. irrational und muss gerundet werden. Dadurch ist es möglich, dass y^{k+1} nicht mehr im relativen Inneren von P_k bzw. der nächste (gerundete) Iterationspunkt x^{k+1} nicht mehr im relativ Inneren von P liegt. Statt K wählt man daher eine kleinere Kugel K' mit dem gleichen Zentrum, so dass auch nach Rundung und Rücktransformation garantiert ist, dass der nächste Iterationspunkt ein relativ innerer Punkt ist.

Analytische Beschreibung eines Iterationsschrittes

Die oben gegebene anschauliche Darstellung wollen wir nun explizit vorführen. Wir starten also mit Problem (5.11). Wir setzen (wie beim Simplexverfahren) voraus, dass A vollen Zeilenrang hat. Außerdem kennen wir mit x^k einen relativ inneren Punkt von P. Ist $D = \operatorname{diag}(x^k)$ die (n,n)-Diagonalmatrix, die die Komponenten $x^k_1, \ldots, x^k_n$ von x^k auf der Hauptdiagonalen enthält, so ist durch

$$T_k(x) \;:=\; \frac{1}{\mathbb{1}^T D^{-1} x}\, D^{-1} x \tag{5.14}$$

eine projektive Transformation definiert. ($T_k(x)$ ist natürlich nur dann endlich, wenn $x \notin \overline{H} = \{y \in \mathbb{R}^n \mid \mathbb{1}^T D^{-1} y = 0\}$ gilt. Wir werden beim Rechnen mit T_k diese Einschränkung nicht weiter erwähnen und gehen davon aus, dass klar ist, wie die Formeln, die T_k enthalten, zu interpretieren sind.) Die projektive Transformation T_k hat, wie leicht zu sehen ist, folgende Eingeschaften:

(5.15) Satz (Eigenschaften von T_k).

(a) $x \geq 0 \implies T_k(x) \geq 0$.

(b) $\mathbb{1}^T x = 1 \implies \mathbb{1}^T T_k(x) = 1$.

(c) $T_k^{-1}(y) = \frac{1}{\mathbb{1}^T D y}\, Dy$ *für alle* $y \in \Sigma$.

(d) $T_k(\Sigma) = \Sigma$.

(e) $P_k := T_k(P) = T_k(\Sigma \cap \Omega) = \Sigma \cap \Omega_k$, *wobei* $\Omega_k = \{y \in \mathbb{R}^n \mid ADy = 0\}$.

(f) $T_k(x^k) = \frac{1}{\mathbb{1}^T\mathbb{1}}\, D^{-1}x^k = \frac{1}{n}\mathbb{1} \in P_k$. △

Die Transformation T_k ist also eine bijektive Abbildung $P \longrightarrow P_k$ und $\Sigma \longrightarrow \Sigma$, die den Punkt x^k in das Zentrum $\frac{1}{n}\mathbb{1}$ von Σ abbildet. Aus (5.15) folgt

$$\begin{array}{rcccl} \min c^T x & = & \min c^T x & = & \min\ c^T T_k^{-1}(y) \\ Ax = 0 & & x \in P & & y \in P_k = T_k(P) \\ \mathbb{1}^T x = 1 & & & & \\ x \geq 0 & & & & \end{array} \quad = \quad \min\ \frac{1}{\mathbb{1}^T Dy}\, c^T Dy \quad \begin{array}{r} \\ ADy = 0 \\ \mathbb{1}^T y = 1 \\ y \geq 0 \end{array} \tag{5.16}$$

Das letzte Programm in (5.16) ist das in der geometrisch-anschaulichen Erläuterung weiter oben genannte Programm mit nichtlinearer Zielfunktion $\frac{1}{\mathbb{1}^T Dy}\, Dy$. Wir linearisieren diese durch Weglassen des Nenners wie folgt:

$$\bar{c}^T := c^T D \tag{5.17}$$

und erhalten das lineare (Hilfs)-Programm

$$\begin{aligned} \min\ & \bar{c}^T y \\ & ADy = 0 \\ & \mathbb{1}^T y = 1 \\ & y \geq 0 \end{aligned} \tag{5.18}$$

Wir vereinfachen (5.18) dadurch, dass wir die Nichtnegativitätsbedingung in (5.18) durch die Bedingung ersetzen, dass y in einer Kugel um $\frac{1}{n}\mathbb{1}$ liegt. Die größte Kugel mit diesem Zentrum, die man dem Simplex Σ einbeschreiben kann, hat den Radius $1/\sqrt{n(n-1)}$. Wie bereits erwähnt, müssen wir aus technischen Gründen eine kleinere Kugel wählen. Es wird sich zeigen, dass man mit der Hälfte dieses optimalen Radius auskommt. Wir betrachten also das Programm

$$\begin{aligned} \min\ & \bar{c}^T y \\ & ADy = 0 \\ & \mathbb{1}^T y = 1 \\ & \left\| y - \tfrac{1}{n}\mathbb{1} \right\| \leq \frac{1}{2}\, \frac{1}{\sqrt{n(n-1)}} \end{aligned} \tag{5.19}$$

Das Minimum von (5.19) kann explizit bestimmt werden.

(5.20) Lemma. *Die Optimallösung von* (5.19) *ist der Vektor*

$$y^{k+1} := \frac{1}{n}\,\mathbb{1} - \frac{1}{2}\, \frac{1}{\sqrt{n(n-1)}}\, \frac{(I - DA^T(AD^2A^T)^{-1}AD - \frac{1}{n}\mathbb{1}\mathbb{1}^T)D\bar{c}}{\|(I - DA^T(AD^2A^T)^{-1}AD - \frac{1}{n}\mathbb{1}\mathbb{1}^T)D\bar{c}\|}. \qquad \triangle$$

Beweis. Zunächst projizieren wir $\bar{c}$ orthogonal auf den linearen Raum $L := \{x \in \mathbb{R}^n \mid ADx = 0, \mathbb{1}^T x = 0\}$. Setzen wir

$$B = \begin{pmatrix} AD \\ \mathbb{1}^T \end{pmatrix},$$

so ist die Projektion von $\bar{c}$ auf L gegeben durch

$$\bar{\bar{c}} = (I - B^T(BB^T)^{-1}B)\,\bar{c},$$

denn offenbar gilt $B\bar{\bar{c}} = 0$, $(\bar{c} - \bar{\bar{c}})^T\bar{\bar{c}} = 0$. Beachten wir, dass $AD\mathbb{1} = Ax^k = 0$ gilt, so folgt durch einfaches Ausrechnen

$$\begin{aligned} BB^T &= \begin{pmatrix} AD^2A^T & AD\mathbb{1} \\ (AD\mathbb{1})^T & \mathbb{1}^T\mathbb{1} \end{pmatrix} = \begin{pmatrix} AD^2A^T & 0 \\ 0 & n \end{pmatrix} \\ (BB^T)^{-1} &= \begin{pmatrix} (AD^2A^T)^{-1} & 0 \\ 0 & \frac{1}{n} \end{pmatrix} \\ B^T(BB^T)^{-1}B &= DA^T(AD^2A^T)^{-1}AD + \frac{1}{n}\mathbb{1}\mathbb{1}^T, \end{aligned}$$

und somit

$$\bar{\bar{c}} = (I - DA^T(AD^2A^T)^{-1}AD - \frac{1}{n}\mathbb{1}\mathbb{1}^T)\,\bar{c}.$$

Für $y \in L$ gilt nach Konstruktion $\bar{c}^T y = \bar{\bar{c}}^T y$ und somit ist das Minimum von (5.19) gleich dem Minimum der Funktion $\bar{\bar{c}}^T y$ unter den Nebenbedingungen von (5.19). Aufgrund unserer Konstruktion ist dieses Minimum gleich dem Minimum von

$$\begin{aligned} &\min\ \bar{\bar{c}}^T y \\ &\left\|y - \tfrac{1}{n}\mathbb{1}\right\| \le \frac{1}{2}\,\frac{1}{\sqrt{n(n-1)}}, \end{aligned}$$

also dem Minimum einer linearen Zielfunktion über einer Kugel. Offenbar wird das Minimum hier durch den Vektor angenommen, den man durch einen Schritt vom Mittelpunkt $\frac{1}{n}\mathbb{1}$ aus in Richtung $-\bar{\bar{c}}$ mit der Länge des Kugelradius erhält, also durch

$$y^{k+1} \;=\; \frac{1}{n}\mathbb{1} - \frac{1}{2}\,\frac{1}{\sqrt{n(n-1)}}\,\frac{1}{\|\bar{\bar{c}}\|}\,\bar{\bar{c}}.$$

Hieraus folgt die Behauptung. □

Damit haben wir ein Minimum von (5.19) analytisch bestimmten können und unser vereinfachtes Hilfsproblem gelöst. Den nächsten Iterationspunkt erhält man durch Anwendung der inversen projektiven Transformation T_k^{-1} auf den in (5.20) bestimmten Vektor y^{k+1}:

$$x^{k+1} \;:=\; T_k^{-1}(y^{k+1}) \;=\; \frac{1}{\mathbb{1}^T D y^{k+1}}\,D y^{k+1}. \tag{5.21}$$

Dem Hilfsprogramm (5.19) kann man übrigens noch eine andere geometrische Interpretation geben. Setzen wir

$$z := Dy, \quad \text{bzw.} \quad y := D^{-1}z,$$

so kann man (5.19) wie folgt schreiben

$$\begin{aligned} \min\ & c^T z \\ & Az = 0 \\ & \mathbb{1}^T D^{-1} z = 1 \\ & z \in E\left(\tfrac{1}{4n(n-1)} D^2, \tfrac{1}{n} x^k\right). \end{aligned} \tag{5.22}$$

Denn wegen (5.17) gilt $\bar{c}^T y = c^T z$ und aus $D^{-1}x^k = \mathbb{1}$ folgt:

$$\begin{aligned} \left\|y - \tfrac{1}{n}\mathbb{1}\right\| \le \frac{1}{2}\frac{1}{\sqrt{n(n-1)}} &\iff \left\|D^{-1}z - \tfrac{1}{n}D^{-1}x^k\right\| \le \frac{1}{2}\frac{1}{\sqrt{n(n-1)}} \\ &\iff \left\|D^{-1}(z - \tfrac{1}{n}x^k)\right\| \le \frac{1}{2}\frac{1}{\sqrt{n(n-1)}} \\ &\iff (z - \tfrac{1}{n}x^k)^T D^{-2}(z - \tfrac{1}{n}x^k) \le \frac{1}{4n(n-1)} \\ &\iff (z - \tfrac{1}{n}x^k)(4n(n-1))D^{-2}(z - \tfrac{1}{n}x^k) \le 1 \\ &\overset{(4.16)}{\iff} z \in E\left(\tfrac{1}{4n(n-1)} D^2, \tfrac{1}{n} x^k\right) \end{aligned}$$

Gehen wir vom Ursprungsproblem (5.11) aus, so bedeutet dies also, dass wir in unserem Hilfsprogramm die Originalzielfunktion c und den linearen Raum Ω unverändert lassen. Wir ersetzen die Hyperebene E durch $E_k = \{z \mid \mathbb{1}^T D^{-1} z = 1\}$ und die Nichtnegativitätsbedingung $x \in \mathbb{R}^n_+$ durch das Ellipsoid $E(\frac{1}{4n(n-1)} D^2, \frac{1}{n} x^k)$ und optimieren hierüber. Aus einer Optimallösung, sagen wir z^{k+1}, von (5.22) erhalten wir eine Optimallösung von (5.19) durch

$$y^{k+1} = D^{-1} z^{k+1}.$$

Wie im Beweis von (5.20) kann man zeigen, dass (5.22) durch eine direkte Formel gelöst werden kann (das folgt natürlich auch aus (5.20) durch die obige Gleichung), und zwar gilt:

$$z^{k+1} \;=\; \frac{1}{n}x^k - \frac{1}{2}\frac{1}{\sqrt{n(n-1)}} \frac{D(I - DA^T(AD^2A^T)^{-1}AD - \frac{1}{n}\mathbb{1}\mathbb{1}^T)Dc}{\|(I - DA^T(AD^2A^T)^{-1}AD - \frac{1}{n}\mathbb{1}\mathbb{1}^T)Dc\|}. \tag{5.23}$$

Hieraus ergibt sich über (5.21) eine neue (einfache) Formel zur Berechnung des nächsten Iterationspunktes.

$$x^{k+1} \;:=\; \frac{1}{\mathbb{1}^T D y^{k+1}} D y^{k+1} \;=\; \frac{1}{\mathbb{1}^T z^{k+1}} z^{k+1}. \tag{5.24}$$

Damit haben wir die wesentlichen Bestandteile eines Iterationsschrittes des Karmarkar-Algorithmus beschrieben und können ihn zusammenfassend darstellen, wobei noch das Abbruchkriterium zu begründen ist.

(5.25) Algorithmus (Karmarkar-Algorithmus).
Eingabe: $A \in \mathbb{Q}^{(m,n)}$ und $c \in \mathbb{Q}^n$. Zusätzlich wird vorausgesetzt, dass $\frac{1}{n}A\mathbb{1} = 0$ und $c^T\mathbb{1} > 0$ gilt.
Ausgabe: Ein Vektor x mit $Ax = 0$, $\mathbb{1}^T x = 1$, $x \geq 0$ und $c^T x \leq 0$ oder die Feststellung, dass kein derartiger Vektor existiert.

1. **Initialisierung.** Setze

$$x^0 := \frac{1}{n}\mathbb{1}$$
$$k := 0$$
$$N := 3n(\langle A\rangle + 2\langle c\rangle - n)$$

2. **Abbruchkriterium.**
   (2.a) Gilt $k = N$, dann hat $Ax = 0$, $\mathbb{1}^T x = 1$, $x \geq 0$, $c^T x \leq 0$ keine Lösung, STOP!
   (2.b) Gilt $c^T x^k \leq 2^{-\langle A\rangle - \langle c\rangle}$, dann ist eine Lösung gefunden. Falls $c^T x^k \leq 0$, dann ist x^k eine Lösung, andernfalls kann wie bei der Ellipsoidmethode (Satz (4.29)) aus x^k ein Vektor $\overline{x}$ konstruiert werden mit $c^T\overline{x} \leq 0$ $A\overline{x} = 0$, $\mathbb{1}^T\overline{x} = 1$, $\overline{x} \geq 0$, STOP!

3. **Update.**
   (3.a) $D := \operatorname{diag}(x^k)$
   (3.b) $\overline{\overline{c}} := (I - DA^T(AD^2A^T)^{-1}AD - \frac{1}{n}\mathbb{1}\mathbb{1}^T)Dc$ (siehe Beweis von (5.20))
   (3.c) $y^{k+1} := \frac{1}{n}\mathbb{1} - \frac{1}{2}\frac{1}{\sqrt{n(n-1)}}\frac{1}{\|\overline{\overline{c}}\|}\overline{\overline{c}}$
   (3.d) $x^{k+1} := \frac{1}{\mathbb{1}^T Dy^{k+1}}Dy^{k+1}$
   (3.e) $k := k + 1$

4. Gehe zu 2. △

Die Geschwindigkeit des Algorithmus hängt natürlich nur von der Implementierung des Schrittes 3 ab. Wir haben hier die aus Lemma (5.20) gewonnene Formel (5.21) für den nächsten Iterationspunkt x^{k+1} gewählt. Man kann x^{k+1} natürlich auch über (5.23), (5.24) bestimmen. Wesentlich ist, dass man eine Update-Formel findet, in der möglichst wenig arithmetische Operationen auftreten.

Es ist offensichtlich, dass die Hauptarbeit von 3 im Schritt (3.b) liegt. Führt man ihn kanonisch aus, so werden $O(n^3)$ Rechenschritte benötigt. Man beachte, dass sich in jedem Schritt nur die Diagonalmatrix D ändert. Diese Tatsache kann man, wie Karmarkar gezeigt hat, ausnutzen, um 3 in $O(n^{2.5})$ Rechenschritten zu erledigen. Die Laufzeit beträgt dann insgesamt $O(n^{2.5}(\langle A\rangle + \langle c\rangle))$.

Wie bei der Ellipsoidmethode können bei der Ausführung des Karmarkar-Algorithmus irrationale Zahlen (durch Wurzelziehen in 3.c) auftreten. Wir sind also gezwungen, alle

Rechnungen approximativ auszuführen. Die dadurch auftretenden Rundungsfehler akkumulieren sich natürlich. Wir müssen also unsere Rundungsvorschriften so einrichten, dass beim Abbruch des Verfahrens eine korrekte Schlussfolgerung gezogen werden kann. Auch das kann man wie bei der Ellipsoidmethode erledigen. Dies sind (auf Abschätzungstechniken der linearen Algebra und Analysis beruhende) Tricks, mit deren Hilfe Verfahren (5.25) in einen polynomialen Algorithmus abgewandelt werden kann. Da insgesamt höchstens N Iterationen durchgeführt werden, folgt

(5.26) Satz. *Die Laufzeit des (geeignet modifizierten) Karmarkar-Algorithmus (5.25) zur Lösung von Problemen des Typs (5.10) ist $O(n^{3.5}(\langle A\rangle + \langle c\rangle)^2)$.* △

Wir wollen im Weiteren die Rundungsfehler vernachlässigen und annehmen, dass wir in perfekter Arithmetik arbeiten. Es bleibt noch zu zeigen, dass Algorithmus (5.25) mit einem korrekten Ergebnis endet.

Zunächst überlegen wir uns, dass alle Punkte x^k im relativ Inneren von $\Omega \cap E \cap \mathbb{R}^n_+$ enthalten sind.

(5.27) Lemma. *Für alle $k \in \{0, 1, \ldots, N\}$ gilt $Ax^k = 0$, $x^k > 0$, $\mathbb{1}^T x^k = 1$.* △

Beweis. Durch Induktion über k! Für $k = 0$ gilt die Behauptung nach Voraussetzung. Für $k+1$ ist $\overline{\overline{c}}$ die Projektion von c auf $\{x \mid ADx = 0, \mathbb{1}^T x = 0\}$, siehe Beweis von (5.20). Daraus folgt $AD\overline{\overline{c}} = 0$, $\mathbb{1}^T\overline{\overline{c}} = 0$. Mithin ergibt sich

$$\begin{aligned}(\mathbb{1}^T Dy^{k+1})Ax^{k+1} = \; ADy^{k+1} \; &= \; AD\left(\frac{1}{n}\mathbb{1} - \frac{1}{2}\,\frac{1}{\sqrt{n(n-1)}}\,\frac{1}{\|\overline{\overline{c}}\|}\,\overline{\overline{c}}\right)\\ &= \; AD\left(\frac{1}{n}\mathbb{1}\right) \; = \; Ax^k = 0.\end{aligned}$$

Daraus folgt $Ax^{k+1} = 0$.

Um $x^k > 0$ zu zeigen, stellen wir zunächst fest, dass

$$K := \left\{y \mid \|y - \frac{1}{n}\mathbb{1}\| \le \frac{1}{2}\,\frac{1}{\sqrt{n(n-1)}}\right\} \subseteq \mathbb{R}^n_+.$$

Denn gibt es ein $\overline{y} \in K$ mit einer nichtpositiven Komponente, sagen wir $\overline{y}_1 \le 0$, so gilt

$$\sum_{i=2}^{n}\left(\overline{y}_i - \frac{1}{n}\right)^2 \le \frac{1}{4n(n-1)} - \left(\overline{y}_1 - \frac{1}{n}\right)^2 \le \frac{1}{4n(n-1)} - \frac{1}{n^2} = \frac{-3n+4}{4n^2(n-1)} < 0,$$

ein Widerspruch. Daraus folgt $y^{k+1} > 0$, und (5.21) ergibt direkt $x^{k+1} > 0$.

Aus (5.21) folgt ebenfalls sofort, dass $\mathbb{1}^T x^{k+1} = 1$ gilt. □

Entscheidend für die Abschätzung der Anzahl der Iterationsschritte ist die folgende Beobachtung.

(5.28) Lemma. *Gibt es ein $x \ge 0$ mit $Ax = 0$, $\mathbb{1}^T x = 1$ und $c^T x \le 0$, dann gilt für alle k:*

$$\frac{(c^T x^{k+1})^n}{\prod_{i=1}^n x_i^{k+1}} \; \le \; \frac{2}{e}\,\frac{(c^T x^k)^n}{\prod_{i=1}^n x_i^k}.$$ △

Wir wollen uns nun überlegen, wieviele Iterationen im Verfahren (5.25) höchstens durchgeführt werden müssen. Der erste Iterationspunkt ist der Vektor

$$x^0 = \frac{1}{n}\mathbb{1}.$$

Daraus folgt mit einer groben Abschätzung aus (4.6)(a)

$$c^T x^0 \;\le\; \frac{1}{n}\sum_{i=1}^{n}|c_i| \;\le\; \frac{1}{n}\sum_{i=1}^{n}2^{\langle c_i\rangle-1} \;<\; \frac{1}{n}2^{\langle c\rangle-n}.$$

Aus Lemma (5.28) ergibt sich (unter den dort gemachten Voraussetzungen)

$$\frac{(c^T x^k)^n}{\prod_{i=1}^n x_i^k} \;\le\; \left(\frac{2}{e}\right)^k \frac{(c^T x^0)^n}{\prod_{i=1}^n x_i^0}$$

und somit wegen $\prod_{i=1}^n x_i^k \le 1$ und $\prod_{i=1}^n x_i^0 = (\frac{1}{n})^n$:

$$c^T x^k \;<\; \left(\frac{2}{e}\right)^{\frac{k}{n}} \left(\frac{\prod x_i^k}{\prod x_i^0}\right)^{\frac{1}{n}} \frac{1}{n}\, 2^{\langle c\rangle-n} \;\le\; \left(\frac{2}{e}\right)^{\frac{k}{n}} 2^{\langle c\rangle-n}. \tag{5.29}$$

In jedem Schritt schrumpft also der Zielfunktionswert um den nur von der Dimension n abhängigen Faktor $\sqrt[n]{\frac{2}{e}}$. Setzen wir

$$N := 3n(\langle A\rangle + 2n\langle c\rangle - n) \tag{5.30}$$

dann gilt:

(5.31) Satz. *Es gibt ein $x \ge 0$ mit $Ax = 0$, $\mathbb{1}^T x = 1$, $c^T x \le 0$ genau dann, wenn es ein $k \in \{0, \ldots, N\}$ gibt mit*

$$c^T x^k < 2^{-\langle A\rangle-\langle c\rangle}.$$

△

Beweis. „$\Longrightarrow$“: Angenommen, es gibt ein $x \in P = \Sigma \cap \Omega$ mit $c^T x \le 0$, dann sind die Voraussetzungen von Lemma (5.28) erfüllt und folglich gilt mit (5.29) und (5.30)

$$c^T x^N \;<\; \left(\frac{2}{e}\right)^{\frac{N}{n}} 2^{\langle c\rangle-n} \;<\; 2^{-\langle A\rangle-\langle c\rangle},$$

wobei wir die Tatsache ausgenutzt haben, dass $3 > \frac{-1}{\log(\frac{2}{e})}$ gilt.

„$\Longleftarrow$“: Angenommen $c^T x^k < 2^{-\langle A\rangle-\langle c\rangle}$ gilt für ein $k \in \{0, \ldots, N\}$. Da $P \neq \emptyset$ ein Polytop ist, wird $\min\{c^T x \mid x \in P\}$ in einer Ecke von P angenommen. Nach Satz (4.11) ist der Optimalwert dieses Programms eine rationale Zahl $\frac{p}{q}$ mit

$$1 \;\le\; q \;\le\; 2^{\langle A\rangle+\langle \mathbb{1}\rangle+\langle c\rangle-n^2-n} \;<\; 2^{\langle A\rangle+\langle c\rangle}.$$

Aus $\frac{p}{q} \le c^T x^k < 2^{-\langle A\rangle-\langle c\rangle}$ folgt dann $\frac{p}{q} \le 0$. □

5.2 Primaler Pfadverfolgungsalgorithmus

Wir skizzieren nun ein Beispiel einer Klasse von Innere-Punkte-Methoden, die primale Pfadverfolgungsalgorithmen genannt werden. Wir folgen dabei der Darstellung dieses Themas in Kapitel 9 von Bertsimas und Tsitsiklis (1997). Weitere Literatur hierzu findet sich in Roos et al. (2006) und Wright (1997).

Die Idee der Algorithmen ist die Folgende. Wir gehen aus von einem LP in Standardform (P) und seinem dualen Programm (D) mit zusätzlich eingeführten Schlupfvariablen s in der folgenden Form:

$$\begin{array}{rl} \min & c^T x \\ & Ax = b \\ & x \geq 0 \end{array} \qquad \text{(P)} \qquad \text{und} \qquad \begin{array}{rl} \max & u^T b \\ & u^T A + s^T = c^T \\ & s \geq 0 \end{array} \qquad \text{(D)}$$

Wir starten mit einem für (P) zulässigen Punkt x^0, dessen Komponenten strikt positiv sind. Dieser Punkt $x^0 \in \mathring{P} := \{x \in \mathbb{R}^n \mid Ax = b, x > 0\}$ ist ein innerer Punkt im Sinne dieser Vorlesung, d. h. ein innerer Punkt in der Relativtopologie des Zulässigkeitsbereichs. Wir versuchen nun eine Folge von Punkten $(x^k)_{k\geq 1}$ zu generieren, die alle in $\mathring{P}$ liegen, die Abstand vom Rand des Polyeders halten und gegen einen Optimalwert konvergieren. Bei unserem speziellen Polyedertyp sind alle Randpunkte in mindestens einer der Mengen $\{x \in \mathbb{R}^n \mid x_i = 0\}$, $i = 1, \ldots, n$ enthalten. Die gewünschte Abstandswahrung vom Rand versuchen wir dadurch zu erreichen, dass eine Annäherung an den Rand in einer Komponente x_i^k bestraft wird. Eine Funktion, die so etwas leistet, ist der Logarithmus.

Für $\mu > 0$ definieren wir die *Barriere-Funktion*

$$B_\mu(x) := \begin{cases} c^T x - \mu \sum_{j=1}^{n} \log x_j & \text{für } x > 0, \\ \infty & \text{falls } x_j \leq 0 \text{ für ein } j = 1, \ldots, n. \end{cases} \tag{5.32}$$

Das *Barriere-Problem* lautet dann:

$$\begin{array}{rl} \min & B_\mu(x) \\ & Ax = b \end{array} \tag{5.33}$$

Diese Definition impliziert, dass alle Punkte, die $Ax = b$ erfüllen, aber nicht in $\mathring{P}$ liegen, einen unendlichen Zielfunktionswert haben. Falls es also Optimallösungen von (5.33) mit endlichem Wert gibt, so sind diese zulässig für (P). Durch wiederholte Lösung von (5.33) bei Veränderung des Barriere-Parameters μ wird versucht, eine Annäherung an eine Optimallösung von (P) zu erreichen.

Die Barriere-Funktion (5.32) ist strikt konvex. Dies impliziert, dass eine Optimallösung mit endlichem Wert, falls eine solche existiert, eindeutig ist. Wir treffen die Annahme, dass alle Barriere-Probleme für $\mu > 0$ eine Optimallösung $x(\mu)$ haben, die somit eindeutig ist. Mit „Standardtricks“ kann man erreichen, dass diese Annahme beim Beginn des Verfahrens erfüllt ist. Beim Simplex-Verfahren haben wir derartige Methoden ausführlich diskutiert, hier verzichten wir darauf.

(5.34) Definition. *Für alle $\mu > 0$ seien Barriere-Probleme* (5.33) *gegeben, deren Optimallösungen jeweils mit $x(\mu)$ bezeichnet seien. Der* zentrale Pfad *ist die Menge $\{x(\mu) \mid \mu > 0\}$ und der (wohldefinierte) Punkt $\lim_{\mu\to\infty} x(\mu)$ heißt* analytisches Zentrum. △

Das analytische Zentrum ist ein Punkt, der – im Sinne der logarithmischen Bestrafungsfunktion – so weit wie möglich von allen Rändern des zulässigen Bereichs entfernt ist, siehe Abbildung 5.2. $\lim_{\mu\to 0} x(\mu)$ ist die Optimallösung unseres originalen LPs.

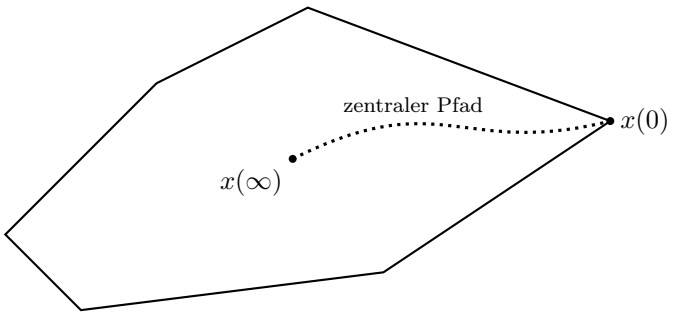


Abbildung 5.2: Zentraler Pfad

(5.35) Beispiel.

(a) Betrachte das Minimierungsproblem

$$\begin{aligned} \min\ & x \\ & x \geq 0 \end{aligned}$$

Die Barriere-Funktion lautet: $B_\mu(x) = x - \mu \log x$, $(x > 0)$. Zum Auffinden des Minimums leiten wir ab:

$$\frac{d}{dx} B_\mu(x) = 1 - \frac{\mu}{x} = 0 \iff x = \mu,$$

Die Optimallösung zum Barriere-Problem lautet also $x(\mu) = \mu$, d.h.

$$\lim_{\mu\to 0} x(\mu) = 0.$$

(b) Betrachte das Minimierungsproblem

$$\begin{aligned} \min\ & x_2 \\ & x_1 + x_2 + x_3 = 1 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Das Barriere-Problem lautet:

$$\begin{aligned} \min\ & x_2 - \mu \log x_1 - \mu \log x_2 - \mu \log x_3 \\ & x_1 + x_2 + x_3 = 1 \end{aligned}$$

Einsetzen der Gleichheitsbedingung in die Zielfunktion führt zu folgendem zentralem Pfad:

$$x(\mu) \;=\; \frac{1}{4}\begin{pmatrix} 1-3\mu+\sqrt{9\mu^2+2\mu+1} \\ 2+6\mu-2\sqrt{9\mu^2+2\mu+1} \\ 1-3\mu+\sqrt{9\mu^2+2\mu+1} \end{pmatrix}$$

mit

$$\lim_{\mu\to 0} x(\mu) = \begin{pmatrix} 1/2 \\ 0 \\ 1/2 \end{pmatrix}, \qquad \lim_{\mu\to\infty} x(\mu) = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix},$$

siehe Abbildung 5.3. An diesem Beispiel sieht man, dass bei diesem Ansatz eine

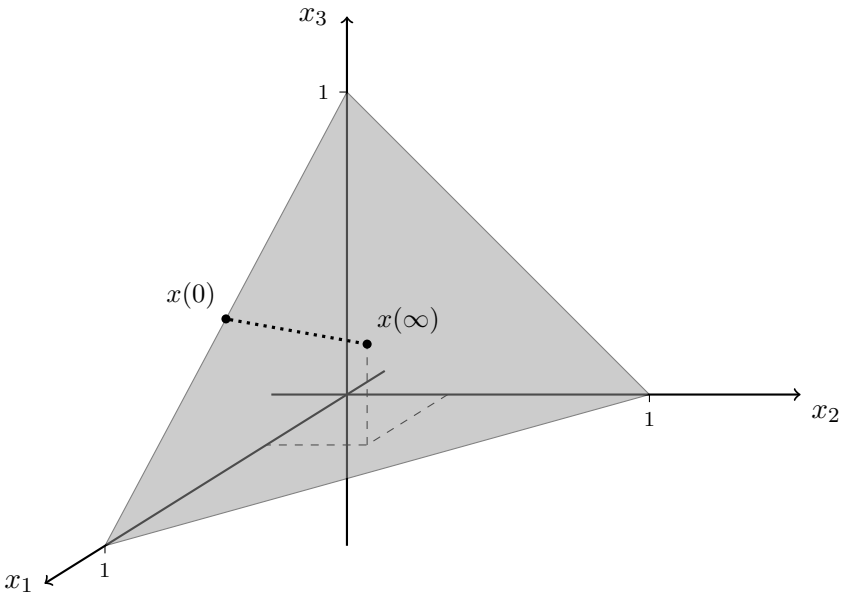


Abbildung 5.3: Beispiel (5.35)(b)

Optimallösung angestrebt wird, die „in der Mitte“ der optimalen Seitenfläche liegt. △

Zum Dualproblem (D) kann man analog ein zugehöriges duales Barriere-Problem definieren:

$$\begin{aligned} \max\; & u^T b + \mu \sum_{j=1}^{n} \log s_j \\ & u^T A + s^T = c^T \end{aligned} \tag{5.36}$$

Wendet man den Kuhn-Tucker-Satz aus der nichtlinearen Optimierung (dieser ist eine Verallgemeinerung des Dualitätssatzes der linearen Optimierung) auf die nichtlinearen Programme (5.33) und (5.36) an, so ergibt sich:

(5.37) Satz (Kuhn-Tucker). *$x^* \in \mathbb{R}^n$, $u^* \in \mathbb{R}^m$ und $s^* \in \mathbb{R}^n$ sind optimal bezüglich* (5.33) *und* (5.36) *genau dann, wenn die folgenden Bedingungen (auch genannt* Karush-Kuhn-Tucker-Bedingungen*) gelten:*

$$\begin{aligned} Ax^* &= b \\ x^* &\geq 0 \\ A^T u^* + s^* &= c \\ s^* &\geq 0 \\ x_j^* s_j^* &= \mu \qquad \forall j = 1, \ldots, n \end{aligned}$$ △

Beweis. Siehe Bertsimas und Tsitsiklis (1997), S. 421. □

Wir nehmen nun an, dass $x \in \mathring{P}$ gilt. Die logarithmische, und damit schwer zu behandelnde primale Barriere-Funktion ersetzen wir für $x > 0$ durch die Approximation durch die ersten Glieder ihrer Taylorreihe.

$$\begin{aligned} B_\mu(x+d) &\approx B_\mu(x) + \sum_{i=1}^n \frac{\partial}{\partial x_i} B_\mu(x) d_i + \frac{1}{2} \sum_{i,j=1}^n \frac{\partial^2}{\partial x_i \partial x_j} B_\mu(x) d_i d_j \\ &= B_\mu(x) + \sum_{i=1}^n \Big(c_i - \frac{\mu}{x_i}\Big) d_i + \frac{1}{2} \sum_{i=1}^n \frac{\mu}{x_i^2} d_i^2 \\ &= B_\mu(x) + (c^T - \mu \mathbb{1}^T X^{-1}) \cdot d + \frac{1}{2}\, \mu d^T (X^2)^{-1} d \end{aligned}$$

mit

$$X := \operatorname{diag}(x) = \begin{pmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{pmatrix}, \qquad d = \begin{pmatrix} d_1 \\ \vdots \\ d_n \end{pmatrix}.$$

Anstatt die Barriere-Funktion direkt zu minimieren, suchen wir eine Richtung d, die diese (abgebrochene) Taylorentwicklung von $B_\mu(x+d)$ minimiert, so dass $x + d$ wieder die Nebenbedingung $A(x+d) = b$, und das heißt $Ad = 0$, erfüllt:

$$\begin{aligned} \min\ & (c^T - \mu \mathbb{1}^T X^{-1}) \cdot d + \frac{1}{2} \mu d^T (X^2)^{-1} d \\ & Ad = 0 \end{aligned} \tag{5.38}$$

Das Problem (5.38) wird nun mit Hilfe von Lagrange-Multiplikatoren gelöst. Wir definieren eine Lagrange-Funktion mit Lagrange-Multiplikatoren $\Lambda^T = (\lambda_1, \ldots, \lambda_m)$ wie folgt:

$$L(d, \Lambda) = (c^T - \mu \mathbb{1}^T X^{-1}) \cdot d + \frac{1}{2} \mu d^T (X^2)^{-1} d - \Lambda^T A d.$$

Eine Lösung finden wir dadurch, dass wir die folgenden Gleichungen lösen:

$$\frac{\partial}{\partial d_i} L(d, \Lambda) = 0 \quad \forall i = 1, \ldots, n, \qquad \frac{\partial}{\partial \lambda_j} L(d, \Lambda) = 0 \quad \forall j = 1, \ldots, m.$$

Es gilt:

$$\begin{pmatrix} \frac{\partial}{\partial d_1} \\ \vdots \\ \frac{\partial}{\partial d_n} \end{pmatrix} L(d,\Lambda) = c - \mu X^{-1}\mathbb{1} + \mu(X^2)^{-1}d - A^T\Lambda, \qquad \begin{pmatrix} \frac{\partial}{\partial \lambda_1} \\ \vdots \\ \frac{\partial}{\partial \lambda_m} \end{pmatrix} L(d,\Lambda) = Ad,$$

was auf das folgende lineare Gleichungssystem führt:

$$\begin{pmatrix} \mu(X^2)^{-1} & -A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} d \\ \Lambda \end{pmatrix} = \begin{pmatrix} \mu X^{-1}\mathbb{1} - c \\ 0 \end{pmatrix}.$$

Die Lösung dieses Gleichungssystems kann man explizit angeben:

$$\begin{aligned} d(\mu) &= \left(I - X^2A^T(AX^2A^T)^{-1}A\right)\left(X\mathbb{1} - \frac{1}{\mu}X^2c\right), \\ \Lambda(\mu) &= \left(AX^2A^T\right)^{-1} A\left(X^2c - \mu X\mathbb{1}\right). \end{aligned}$$

Hat man eine zulässige Lösung zum aktuellen Barriere-Problem (5.33) für ein bestimmtes μ, so bekommt man auf diese Weise eine neue zulässige Lösung $x + d(\mu)$ für (5.33). Der Vektor $d(\mu)$ heißt *Newton-Richtung*, die obige Berechnung ein *Newton-Schritt*, da er als Schritt in der Anwendung des Newton-Verfahrens der nichtlinearen Optimierung aufgefasst werden kann. Iteriert man dieses Vorgehen, dann konvergiert die Folge der so erzeugten Lösungen gegen die (beweisbar eindeutige) Optimallösung $x(\mu)$ des Barriere-Problems (5.33) zum Parameter μ.

Um das Originalproblem zu lösen, bedient man sich dieser Idee, geht aber etwas anders vor: Man verringert μ in jeder Iteration durch Multiplizieren mit einem festen Parameter α, wobei $0 < \alpha < 1$. Ein allgemeiner Schritt im Gesamtverfahren, das mit einem Startpunkt $x^0 \in \mathring{P}$, einem Barriere-Parameter $\mu > 0$ und einem Skalierungsparameter $0 < \alpha < 1$ startet, sieht dann wie folgt aus (hierbei ist $k = 0, 1, 2, \ldots$ der Iterationszähler): Wir haben einen gegenwärtigen Punkt $x^k \in \mathring{P}$ und den aktuellen Barriere-Parameter $\alpha^k\mu$. (Achtung! In α^k ist k als Exponent zu verstehen, wohingegen k an den Variablen x, u und s in diesem Abschnitt als oberer Index steht.) Wenn wir von x^k ausgehend den Punkt $x(\alpha^k\mu)$ (die Lösung des Barriere-Problems (5.33) zum Parameter $\alpha^k\mu$) berechnen wollten, würden wir eine Folge von Newton-Schritten machen, die gegen $x(\alpha^k\mu)$ konvergiert. Wir machen jedoch nur den ersten Schritt dieser Folge und bestimmen von x^k ausgehend die erste Näherungslösung $x^{k+1} := x^k + d(\alpha^k\mu)$ von $x(\alpha^k\mu)$. Statt nun weiter auf $x(\alpha^k\mu)$ zuzusteuern, verändern wir den Barriere-Parameter von $\alpha^k\mu$ auf $\alpha^{k+1}\mu$. Nun bestimmen wir von x^{k+1} ausgehend die erste Näherungslösung $x^{k+2} := x^{k+1} + d(\alpha^{k+1}\mu)$ von dem Punkt $x(\alpha^{k+1}\mu)$ auf dem zentralen Pfad, usw., siehe Abbildung 5.4.

Mit dieser Interpretation ist klar, dass wir ein Verfahren vor uns haben, das versucht, dem zentralen Pfad zum Optimalpunkt zu folgen, aber (zur Aufwandsreduzierung) jeweils nur Punkte in der Nähe des zentralen Pfades erzeugt.

In jedem Schritt k erhält man übrigens auch eine passende duale Lösung zu $x^{k+1} = x^k + d(\alpha^k\mu)$ durch

$$\begin{aligned} u^{k+1} &= \Lambda(\alpha^k\mu), \\ s^{k+1} &= c - A^Tu^{k+1}. \end{aligned}$$

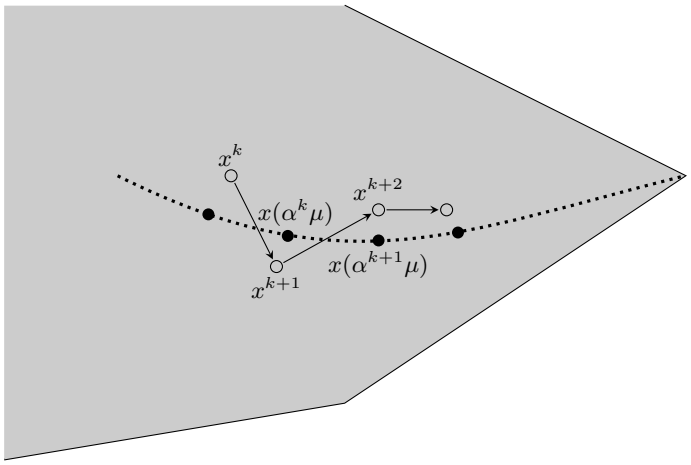


Abbildung 5.4: Newton-Schritte

Wir fassen die Überlegungen zusammen:

(5.39) Algorithmus (Primaler Pfadverfolgungsalgorithmus).
Eingabe: A, b, c, Genauigkeitsparameter $\varepsilon > 0$, $0 < \alpha < 1$, Startparameter $\mu^0 > 0$, zulässige primale und duale Startlösungen $x^0 > 0$, $s^0 > 0$, u^0
Ausgabe: ε-genaue primale und duale Lösungen x, s, u

1. Setze $k := 0$

2. (Optimalitätstest)
   Falls $(s^k)^T x^k < \varepsilon$, setze $x := x^k$, $u := u^k$, $s := s^k$. STOP

3. (Update Barriere-Parameter)
   Setze
   $$X_k := \begin{pmatrix} x_1^k & & \\ & \ddots & \\ & & x_n^k \end{pmatrix}, \qquad \mu^{k+1} := \alpha\mu^k$$

4. (Berechnung Newton-Richtung)
   Löse das lineare Gleichungssystem
   $$\begin{pmatrix} \mu^{k+1}(X_k^2)^{-1} & -A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} d \\ \Lambda \end{pmatrix} = \begin{pmatrix} \mu^{k+1} X_k^{-1}\mathbb{1} - c \\ 0 \end{pmatrix}.$$
   in den Variablen d und Λ.

5. (Update Lösungen)
   Setze

$$\begin{aligned} x^{k+1} &:= x^k + d, \\ u^{k+1} &:= \Lambda, \\ s^{k+1} &:= c - A^T\Lambda. \end{aligned}$$

6. Setze $k := k + 1$ und gehe zu Schritt 2. △

Zur Konvergenz: Man kann zeigen, dass während des Algorithmus gilt

$$\sum_{i=1}^{n}(x_i^k s_i^k - \mu^k)^2 \leq (\mu^k)^2\beta$$

für einen festen Wert $\beta < 1$. Wenn also μ^k nahe genug an 0 ist, muss $x_i^k s_i^k$ nahe an 0 sein für alle $i = 1, \ldots, n$ und die Bedingung für den komplementären Schlupf ist näherungsweise erfüllt. Genauer gilt:

(5.40) Satz. *Für $0 < \beta < 1$ sei*

$$\alpha \;=\; 1 - \frac{\sqrt{\beta} - \beta}{\sqrt{\beta} + \sqrt{n}},$$

$\mu^0 > 0$, $\varepsilon > 0$, und $x^0 > 0$, $s^0 > 0$, u^0 seien zulässige Startlösungen mit

$$\left\| \frac{1}{\mu^0} X_0 S_0 \mathbb{1} - \mathbb{1} \right\| \leq \beta,$$

wobei

$$X_0 \;=\; \begin{pmatrix} x_1^0 & & \\ & \ddots & \\ & & x_n^0 \end{pmatrix}, \qquad S_0 \;=\; \begin{pmatrix} s_1^0 & & \\ & \ddots & \\ & & s_n^0 \end{pmatrix}.$$

Dann findet Algorithmus (5.39) eine primal-duale Lösung x^K, s^K, u^K mit Genauigkeitslücke $(s^K)^T x^K \leq \varepsilon$ nach

$$K = \left\lceil \frac{\sqrt{\beta} + \sqrt{n}}{\sqrt{\beta} - \beta} \; \log \frac{(s^0)^T x^0 (1 + \beta)}{\varepsilon(1 - \beta)} \right\rceil$$

Iterationen. △

Beweis. Siehe Bertsimas und Tsitsiklis (1997), S. 427–429. □

Um das Verfahren zu initialisieren, braucht man noch Startlösungen. Man bekommt zulässige Lösungen $x^0 > 0$, $s^0 > 0$ mit

$$\left\| \frac{1}{\mu^0} X_0 S_0 \mathbb{1} - \mathbb{1} \right\| \leq \beta = \frac{1}{4}$$

aus einem Hilfs-LP, das sich mit „offensichtlicher“ Startlösung lösen lässt, ähnlich wie in Phase I des Simplexalgorithmus.

Nach Satz (5.40) beläuft sich mit $\beta = \frac{1}{4}$ die Anzahl an notwendigen Iterationen, um die Genauigkeitslücke von $\varepsilon_0 = (s^0)^T x^0$ auf ε zu verkleinern, auf

$$\left\lceil (2 + 4\sqrt{n}) \log \left(\frac{\varepsilon_0}{\varepsilon} \cdot \frac{5}{3} \right) \right\rceil,$$

wobei in jeder Iteration $O(n^3)$ Operationen anfallen (Lösung des Gleichungssystems mit $m \leq n$). Damit ist die Gesamtlaufzeit von Algorithmus (5.39) in

$$O\Big(n^3 \sqrt{n} \log \frac{\varepsilon_0}{\varepsilon} \Big).$$

5.3 Primal-dualer Pfadverfolgungsalgorithmus

Die heute verwendeten Barriere-Methoden bauen fast alle auf dem primal-dualen Pfadverfolgungsalgorithmus auf. Die Grundform dieses Verfahrens stammt u. a. von Kojima et al. (1989). In diesem Abschnitt wird das Grundprinzip eines solchen Algorithmus skizziert. Für mehr Details siehe z. B. Wright (1997).

Die Idee besteht darin, die KKT-Bedingungen in (5.37) mit Hilfe des Newton-Verfahrens direkt zu lösen. Ganz allgemein bestimmt das Newton-Verfahren für eine gegebene Funktion $F : \mathbb{R}^r \to \mathbb{R}^r$ eine Nullstelle z^* von F, d. h. ein $z^* \in \mathbb{R}^r$ mit $F(z^*) = 0$, und funktioniert wie folgt: Angenommen, wir haben bereits eine Näherung z für z^*, dann suchen wir eine Richtung $d \in \mathbb{R}^r$, so dass $z + d$ eine bessere Näherung für z^* ist. Approximation von F durch die Taylorentwicklung ergibt wieder:

$$F(z + d) \approx F(z) + J_F(z) \cdot d$$

mit der Jacobi-Matrix

$$J_F(z) \; = \left(\frac{\partial}{\partial z_j} F_i(z) \right)_{1 \leq i,j \leq r}.$$

Das Newton-Verfahren besteht darin, in jedem Schritt ein d (die *Newton-Richtung*) zu suchen, so dass $F(z) + J_F(z) \cdot d = 0$.

In unserem Fall wird die Funktion F durch die Karush-Kuhn-Tucker-Bedingungen bestimmt: ein Punkt $(x^*, u^*, s^*) \in \mathbb{R}^{2n+m}$ mit $x^* \geq 0$ und $s^* \geq 0$ sowie

$$\begin{aligned} Ax^* &= b \\ A^T u^* + s^* &= c \\ X^* S^* \mathbb{1} &= \mu \mathbb{1} \end{aligned}$$

(vgl. (5.37)) ist eine Nullstelle der Funktion

$$F : \mathbb{R}^{2n+m} \to \mathbb{R}^{2n+m}, \qquad F(x, u, s) \; = \begin{pmatrix} Ax - b \\ A^T u + s - c \\ XS\mathbb{1} - \mu \mathbb{1} \end{pmatrix},$$

wobei wieder $X^* := \mathrm{diag}(x^*)$, $S^* := \mathrm{diag}(s^*)$. Zu lösen ist dann in jedem Newton-Schritt das lineare Gleichungssystem

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{pmatrix} \cdot \begin{pmatrix} d_x \\ d_u \\ d_s \end{pmatrix} = -F(x,u,s) = \begin{pmatrix} 0 \\ 0 \\ \mu \mathbb{1} - XS\mathbb{1} \end{pmatrix}, \tag{5.41}$$

falls die aktuelle Lösung (x, u, s) primal und dual zulässig ist.

Das Newton-Verfahren verwendet keine Schrittlängen. Bei dem hier skizzierten Ansatz kann es daher passieren, dass der nächste Iterationspunkt $(x^{k+1}, u^{k+1}, s^{k+1})$ nicht zulässig ist. Deswegen müssen Schrittlängen eingebaut werden. Es hat sich als sinnvoll erwiesen, für primale und duale Variablen unterschiedliche Schrittlängen β_P und β_D zu wählen. Die neue Lösung ergibt sich dann aus der aktuellen Lösung (x^k, u^k, s^k) durch

$$x^{k+1} = x^k + \beta_P d_x\ , \quad u^{k+1} = u^k + \beta_D d_u\ , \quad s^{k+1} = s^k + \beta_D d_s.$$

Die Nichtnegativitätsbedingung $x^k + \beta_P d_x \geq 0$ übersetzt sich in

$$\beta_P \leq -\frac{x_i}{(d_x)_i} \qquad \forall i \text{ mit } (d_x)_i < 0$$

(analog für β_D), so dass man für die Schrittlängen

$$\beta_P = \min\left\{1, \alpha \min_{\{i|(d_x)_i<0\}} \Big(-\frac{x_i}{(d_x)_i}\Big)\right\},$$
$$\beta_D = \min\left\{1, \alpha \min_{\{i|(d_s)_i<0\}} \Big(-\frac{s_i}{(d_s)_i}\Big)\right\},$$

mit $0 < \alpha < 1$ wählen kann. ($\alpha < 1$, um nicht auf den Rand des Zulässigkeitsbereichs zu stoßen und numerische Probleme zu vermeiden.)

Wie im primalen Pfadverfolgungsalgorithmus wird auch der Barriere-Parameter in jedem Schritt angepasst, z. B. durch

$$\mu^k = \rho_k \frac{(x^k)^T s^k}{n} \tag{5.42}$$

mit $0 < \rho_k \leq 1$. Für die Wahl von ρ_k gibt es verschiedene Möglichkeiten; häufig wird z. B. $\rho_k = 1$ gewählt, solange der Algorithmus relativ große Fortschritte macht, und $\rho_k < 1$ sonst.

(5.43) Algorithmus (Primal-dualer Pfadverfolgungsalgorithmus).

Eingabe: A, b, c, Genauigkeitsparameter $\varepsilon > 0$, zulässige primale und duale Startlösungen $x^0 > 0$, $s^0 > 0$, u^0

Ausgabe: ε-genaue primale und duale Lösungen x, s, u

1. Setze $k := 0$
2. (Optimalitätstest)
   Falls $(s^k)^T x^k < \varepsilon$, setze $x := x^k$, $u := u^k$, $s := s^k$. STOP

3. (Berechnung Newton-Richtung)
   Für ein geeignetes $\rho_k \in (0,1]$ setze

$$\mu^k := \rho_k \frac{(x^k)^T s^k}{n}, \qquad X_k := \begin{pmatrix} x_1^k & & \\ & \ddots & \\ & & x_n^k \end{pmatrix}, \qquad S_k = \begin{pmatrix} s_1^k & & \\ & \ddots & \\ & & s_n^k \end{pmatrix}$$

   und löse das lineare Gleichungssystem

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S_k & 0 & X_k \end{pmatrix} \cdot \begin{pmatrix} d_x^k \\ d_u^k \\ d_s^k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \mu^k \mathbb{1} - X_k S_k \mathbb{1} \end{pmatrix}.$$

4. (Bestimmung Schrittlängen)
   Setze

$$\beta_P^k = \min\left\{1, \alpha \min_{\{i|(d_x^k)_i<0\}} \Big(- \frac{x_i^k}{(d_x^k)_i} \Big) \right\},$$
$$\beta_D^k = \min\left\{1, \alpha \min_{\{i|(d_s^k)_i<0\}} \Big(- \frac{s_i^k}{(d_s^k)_i} \Big) \right\}.$$

5. (Update Lösungen)
   Setze

$$x^{k+1} := x^k + \beta_P d_x^k,$$
$$u^{k+1} := u^k + \beta_D d_u^k,$$
$$s^{k+1} := s^k + \beta_D d_s^k.$$

6. Setze $k := k+1$ und gehe zu Schritt 2. △

Bezüglich der Komplexität des Algorithmus kann man zeigen, dass die Anzahl Iterationen bei Start-Genauigkeit $\varepsilon_0 = (s^0)^T x^0$ im worst case in $O(\sqrt{n} \log \frac{\varepsilon_0}{\varepsilon})$ ist. In der Praxis wurde sogar eine durchschnittliche Laufzeit von $O(\log n \log \frac{\varepsilon_0}{\varepsilon})$ beobachtet, aber bisher nicht bewiesen.

Mehrotras Prediktor-Korrektor-Methode

Die heute in kommerziellen Codes meist verwendeten Barriere-Verfahren sind üblicherweise Varianten der „Prediktor-Korrektor-Methode“ von Merothra, siehe Mehrotra (1992), die im Prinzip folgendermaßen funktionert. In jeder Iteration wird die nächste Richtung mittels Lösen des Gleichungssystems (5.41) bestimmt, wobei jedoch statt μ ein skalierter Wert $\sigma\mu$ benutzt wird. Der Skalierungsfaktor $0 < \sigma < 1$ wird vorher bestimmt, indem ein Newton-Schritt „simuliert“ wird (ohne die primale und duale Gültigkeit von x, u, s vorauszusetzen) und σ je nach Länge des erlaubten Schrittes gewählt wird.

Geometrische Interpretation: Der „simulierte“ Schritt berechnet eine Richtung, in die man gehen möchte (Prediktor-Schritt). Wenn sich diese Richtung als „gut“ erweist, d. h. man durch (5.42) eine deutliche Verringerung von μ erreichen würde, wird σ nahe an 0 gewählt, ansonsten nahe an 1. Danach wird die tatsächliche Richtung unter Verwendung von σ berechnet (Korrektor-Schritt). In der Regel führt dieses Vorgehen zu einer deutlichen Verringerung der Anzahl an Iterationen. Der zusätzliche Aufwand in jedem Schritt hält sich in Grenzen: es ist zwar ein zusätzliches Gleichungssystem zu lösen, dessen linke Seite stimmt aber mit dem ohnehin zu lösenden überein, so dass man die Faktorisierung der Matrix wiederverwenden kann.

Die Prediktor-Korrektor-Methode liefert keine neuen theoretischen Erkenntnisse, hat sich aber in der Praxis als das bisher effizienteste Verfahren zur Lösung von großen LPs herausgestellt. Natürlich gibt es auch hier wieder viele verschiedene Varianten und es sind eine Menge numerischer Details zu beachten, auf die hier nicht näher eingegangen werden kann; für mehr Informationen siehe Wright (1997).

Aussagen führender LP-Software-Anbieter

Im Frühjahr 2013 habe ich einige der führenden kommerziellen Anbieter von Optimierungssoftware gefragt, welche Version der Innere-Punkte- bzw. Barriere-Methoden in ihrem LP-Paket implementiert ist. Bob Bixby (Gurobi) antwortete:

> „We use a primal-dual log-barrier long-path method with predictor corrector, where two important practical issues are the choice of starting point and the handling of dense columns. And, for these last two issues there really isn't any clear winner to point to. Gurobi does also offer the option to run the homogeneous algorithm.... This version has theoretical advantages when handling infeasibilities, though, as it turns out, the standard algorithm can also be slightly modified to produce infeasibility proofs.“

Christian Bliek (IBM, software package CPLEX) schrieb:

> „For lp/qp our barrier solver uses Mehrotra predictor corrector, and Gondzio multiple corrections. We recently switched to the homogeneous method for our default, as I found it to be numerically more robust. For this I had to work a bit to reduce the extra cost associated with this method. We still work with the normal equations, and the normal matrix is factorized using Cholesky. We have three orderings; approximate minimum degree, approximate minimum fill or nested dissection. Our default automatically selects what it thinks the best ordering algorithm is for the problem at hand. Dense columns and free variables are handled in a special way. The challenge for these is numerical stability. Implementers will typically not share too much information about this, but you can find some approaches described in “Primal-Dual Interior-Point Methods” by Stephen Wright or in papers by Csaba Meszaros or Erling Andersen.“

Wie in der Vorlesung mehrfach erwähnt, die Implementierung von Barriere-Methoden ist nicht-trivial, viele theoretisch interessante Varianten sind verfügbar, welche Kombination von welchen „Tricks“ in der Praxis erfolgreich ist, ist theoretisch nicht entscheidbar. Hier weisen nur Rechenexperimente mit relevanten Praxisdaten den Weg. Testläufe mit zufällig erzeugten Daten sind wenig hilfreich.

Christian Blieck hat noch eine Zusatzbemerkung zum Vergleich zwischen dem CPLEX barrier solver und dem von CPLEX angebotenen Simplexverfahren angefügt:

> „On our lp testset barrier beats simplex when the solve times are above 10 sec. Break even is even a bit less than that. One of the advantages of barrier is that it can take advantage of parallelism, while simplex doesn't.“

Ähnliche Aussagen machen auch alle anderen führenden Anbieter von Optimierungssoftware. Dies gilt für das „reine Lösen“ von LPs. Wenn aber ein LP-Löser als Unterroutine in ein Verfahren zur Lösung von MIPs eingebunden ist, sieht die Sache anders aus. Dieser Aspekt wird in den nächsten beiden Kapiteln behandelt.

Literaturverzeichnis

D. Bertsimas und J. N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, 1997.

N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

M. Kojima, S. Mizuno, und A. Yoshise. A primal-dual interior point method for linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior Points and Related Methods*, S. 29–47. Springer Verlag, New York, 1989.

S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2:575–601, 1992.

C. Roos, T. Terlaky, und J.-P. Vial. *Interior point methods for linear optimization.* Springer Verlag, 2nd edition, 2006.

D. Shanno. Who invented the interior-point method? In M. Grötschel, editor, *Optimization Stories*, DOCUMENTA MATHEMATICA, Journal der Deutschen Mathematiker-Vereinigung, S. 55–64. DMV, Bielefeld, 2012. Online: `www.math.uni-bielefeld.de/documenta/vol-ismp/20_shanno-david.pdf`.

S. J. Wright. *Primal-Dual Interior-Point Methods.* SIAM, Philadelphia, 1997.

6 Branch & Bound-Verfahren

Wir wollen uns nun mit ganzzahliger und gemischt-ganzzahliger Optimierung beschäftigen und werden eine Verfahrensklasse angeben, mit der man derartige Probleme exakt lösen kann. Sie basiert auf der Strategie, alle möglichen Lösungen auf geschickte Weise zu enumerieren und durch Benutzung von Primal- und Dualheuristiken den Rechenaufwand zu beschränken.

Die generelle Idee dieser sogenannten Branch & Bound-Verfahren ist ganz einfach. Wir betrachten ein Minimierungsproblem. Man benutze zunächst eine Primal- und eine Dualheuristik (Berechnung von Bounds), sind die Werte der gefundenen Lösung gleich, ist man fertig. Falls nicht, wird die Lösungsmenge partitioniert (Branching), und die Heuristiken werden auf die Teilmengen angewandt. Ist für eine (oder mehrere) dieser Teilmengen die für sie berechnete untere Schranke nicht kleiner als die beste bisher gefundene obere Schranke, braucht man die Lösungen in dieser Teilmenge nicht mehr zu betrachten (man erspart sich also die weitere Enumeration). Ist die untere Schranke kleiner als die beste gegenwärtige obere Schranke, muss man die Teilmenge weiter zerteilen. Man fährt so lange mit der Zerteilung fort, bis für alle Lösungsteilmengen die untere Schranke mindestens so groß ist wie die (global) beste obere Schranke.

Techniken dieser Art erscheinen in der Literatur unter einer Vielzahl von Namen wie z. B.:

- Branch & Bound-Verfahren,
- Verfahren der implizierten Enumeration,
- Divide-and-Conquer-Methoden,
- Backtracking-Methoden,
- Zerlegungs-Verfahren, etc.

Durch geeignete Darstellung kann man zeigen, dass jede dieser Techniken ein Spezialfall der übrigen ist, d. h. im Prinzip tun diese Verfahren alle das gleiche, sie unterscheiden sich nur bezüglich der jeweils angewendeten Strategie bzw. Philosophie. Manche Autoren machen sehr feine Unterschiede zwischen diesen Methoden, wir wollen jedoch alle Verfahren, die Enumerationstechniken benutzen, *Branch & Bound-Algorithmen* nennen.

Das Prinzip der Branch & Bound-Verfahren ist trivial, theoretisch sind sie nicht sonderlich interessant, aber es scheint so, dass man ohne sie einfach nicht auskommt. Bei den meisten in der Praxis erfolgreichen Methoden zur Lösung schwieriger kombinatorischer oder ganzzahliger Optimierungsprobleme wird irgendwann einmal Branch & Bound eingesetzt.

Noch eine Warnung! So simpel die Idee ist, so schwierig ist es i. A. Branch & Bound-Methoden effizient zu implementieren. Eine Hauptschwierigkeit besteht darin, „gute“ Zerlegungstechniken und einfache Datenstrukturen zu erfinden, die die effiziente Abarbeitung und Durchsuchung der einzelnen Teilmengen ermöglichen. Ganz allgemein kann man diesen Algorithmentyp formalisiert wie folgt darstellen.

(6.1) Algorithmus (Allgemeines Branch & Bound-Verfahren).

Eingabe: Eine Menge L von zulässigen Lösungen (implizit gegeben) und eine Zielfunktion $c : L \to \mathbb{K}$, die jedem $S \in L$ einen Wert $c(S)$ zuordnen.

Ausgabe: Lösung von

$$\max_{S \in L} c(S). \tag{P}$$

Annahme: Wir nehmen an, dass es ein „Universum“ $\overline{L}$ gibt mit $L \subseteq \overline{L}$, dass $c(S)$ wohldefiniert ist für alle $S \in \overline{L}$ und dass das relaxierte Problem

$$\max_{S \in \overline{L}} c(S) \tag{RP}$$

„einfach“ lösbar ist. Ferner soll die Lösung von (RP) in „nicht-trivialem“ Zusammenhang mit der Lösung von (P) stehen. Wir nehmen also an, dass wir P relaxieren können und (durch die Lösung von (RP)) eine effiziente Dualheuristik zur Verfügung steht.

1. Berechne eine untere Schranke U für (P). Man wende z. B. eine Primalheuristik zur Auffindung einer zulässigen Lösung an oder setze $U := -\infty$.

2. Setze $\mathcal{K} = \{L\}$ ($\mathcal{K}$ ist die Menge der Kandidatenmengen).

3. Falls $\mathcal{K} = \emptyset \longrightarrow$ STOP (die beste gegenwärtige Lösung ist eine Optimallösung von (P)).
   Andernfalls wähle eine Menge $M \in \mathcal{K}$. (Diesen Schritt nennt man *Branching* oder *Verzweigung*. Man muss sich hier genau überlegen, welche der noch nicht bearbeiteten Kandidatenmengen untersucht werden soll.)

4. **Bounding:** Wähle eine „geeignete“ Menge $\overline{M} \subseteq \overline{L}$ mit $M \subseteq \overline{M}$, so dass das relaxierte Problem

   $$\max_{S \in \overline{M}} c(S) \tag{RPM}$$

   „einfach“ zu lösen ist. (Das auf M eingeschränkte Ursprungsproblem wird also relaxiert. Wegen $M \subseteq \overline{M}$ bildet der Wert $c(S^*)$ der Optimallösung S^* von (RPM) eine obere Schranke (*Bound*) für den Wert der Optimallösung von

   $$\max_{S \in M} c(S) \tag{PM}$$

5. Ist $c(S^*) \leq U$, so hat keine Lösung aus $\overline{M}$ und somit auch keine aus M einen besseren Wert als die beste bereits bekannte Lösung von (P), d. h. die Lösungen aus M brauchen nicht weiter betrachtet zu werden. Entferne also M aus $\mathcal{K}$ und gehe zu 3.

6. Ist $S^* \in L$ und $c(S^*) > U$, so ist eine zulässige Lösung für (P) gefunden, deren Wert besser als U ist. Da die beste Lösung aus M gefunden ist, braucht M nicht mehr weiter untersucht zu werden. Entferne M aus $\mathcal{K}$, setze $U := c(S^*)$ und gehe zu 3.

7. **Separation** (Zerlegung): Es gilt also $c(S^*) > U$ und $S^* \notin L$. In diesem Falle war die Berechnung der unteren Schranke nutzlos, da wir keine verwertbare Aussage machen können. Wir zerlegen daher M in „geeignete“ kleinere Mengen $M_1, M_2, \ldots, M_k$, entfernen M aus $\mathcal{K}$, fügen $M_1, M_2, \ldots, M_k$ zu $\mathcal{K}$ hinzu und gehen zu 3. △

Falls M in den Schritten 5 oder 6 verworfen wird, so sagt man, dass M *ausgelotet* (fathomed) worden ist.

Ist man in Schritt 7 angelangt, so ist es häufig nützlich zu versuchen, aus S^* eine zulässige Lösung S von M zu konstruieren, um damit (durch eine Primalheuristik) die untere Schranke verbessern zu können.

Branch & Bound-Verfahren unterscheiden sich vor allem durch die Wahl verschiedener Relaxierungen $\overline{L}$, durch unterschiedliche Zerlegungsstrategien in 7. und durch die Auswahl der nächsten Kandidatenmenge in 3. Mit jedem Branch & Bound-Verfahren kann man einen sogenannten *Branch & Bound-Baum* assoziieren, der über die Verzweigungen Auskunft gibt. Ein derartiger Verzweigungsbaum ist in Abbildung 6.1 gezeigt.

Bei der Lösung des Teilproblems 1 wird eine obere Schranke mit Wert 20 gefunden und durch eine Heuristik eine zulässige Lösung von (P) mit Wert 7. Links vom jeweiligen Knoten ist die Reihenfolge vermerkt, in welcher die Teilprobleme gelöst wurden, rechts neben den Knoten die jeweils berechnete obere Schranke. Bei der Lösung des 6. Teilproblems wird eine zulässige Lösung mit Wert 10 gefunden. Das 15. Teilproblem liefert die Optimallösung.

Das Hauptproblem bei der Implementierung von Branch & Bound-Verfahren besteht darin, dass man sich alle Kandidatenmengen merken muss. Dies muss auf geschickte Weise geschehen, um nicht zu viel suchen zu müssen und nicht zu viel Speicherplatz zu vergeuden. Außerdem muss man dafür Sorge tragen, dass der Baum nicht zu groß wird und der Speicherplatz ausreicht. Man muss also in vielerlei Hinsicht Kompromisse schließen, um ein praktikables Verfahren zu entwerfen.

Das Branch & Bound-Verfahren ist so organisiert, dass in jedem Schritt folgendes gilt:

> Eine zulässige Lösung S gehört entweder zu einer bereits verworfenen Menge oder sie ist in genau einer der Kandidatenmengen enthalten.

Falls also die Menge der zulässigen Lösungen endlich ist und jedes Subproblem (RPM) in endlicher Zeit gelöst wird, bricht das Verfahren nach endlich vielen Schritten ab, wenn wir die (sinnvolle) Konvention treffen, dass einelementige Mengen M nicht relaxiert werden.

Wir wollen uns nun überlegen, wie wir die Branch & Bound-Technik bei ganzzahligen Programmierungsproblemen einsetzen können.

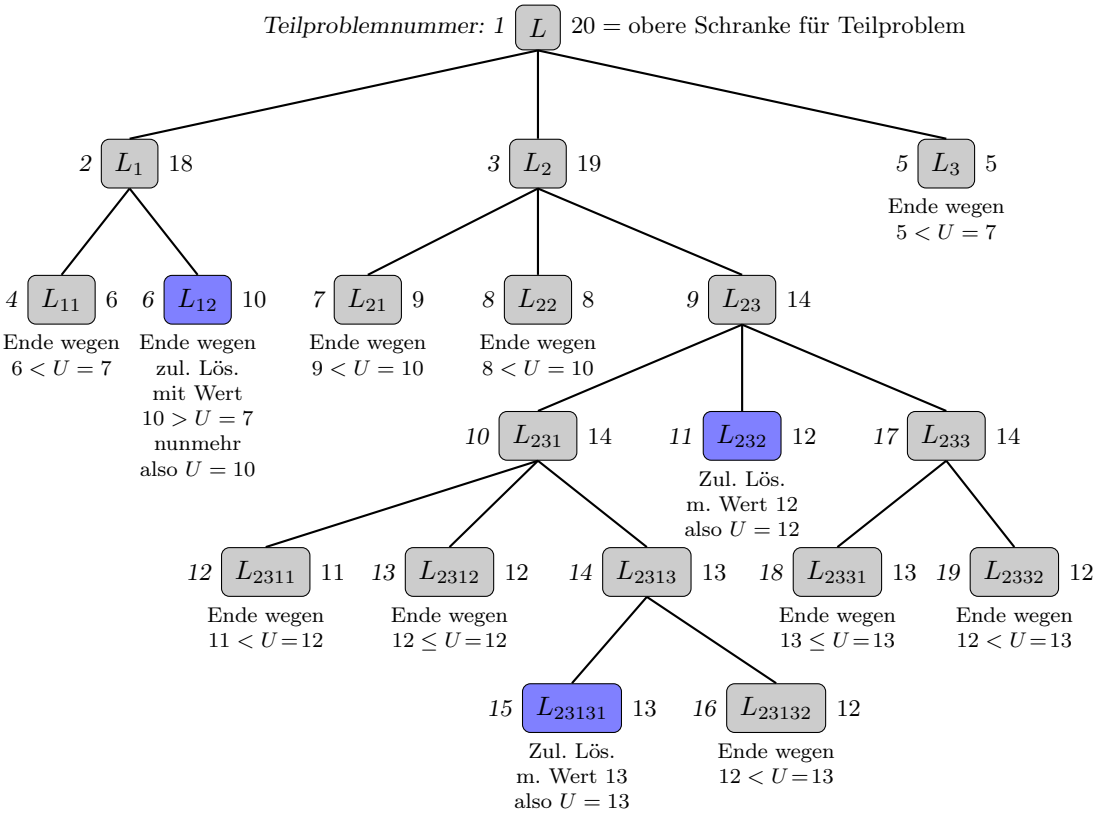


Abbildung 6.1: Ein Branch & Bound-Baum. Der Baum wird in der Reihenfolge der Problemnummern durchlaufen. Zu Beginn hat die untere Schranke U den Wert 7, in den farbigen Knoten wird U erhöht. Die Optimallösung (mit Wert 13) wird im Knoten L_{23131} gefunden.

(6.2) Algorithmus (Branch & Bound-Verfahren von Dakin für gemischt-ganzzahlige Programme).

Eingabe: Gemischt-ganzzahliges Programm mit rationalen Daten

$$\begin{aligned} \max\ & c^T x \\ & Ax = b \\ & x \geq 0 \\ & x_i \text{ ganzzahlig für alle } i \in N_1 \end{aligned} \tag{MIP$^=$}$$

$$P_0 := \{x \mid Ax = b, x \geq 0, x_i \text{ ganzzahlig für alle } i \in N_1\}$$

Ausgabe: Lösung von (MIP$^=$)

1. Setze:

untere Schranke	$U := -\infty$
Kandidatenmengen	$\mathcal{K} := \{P_0\}$
Zählindex	$k := 0$
gegenwärtig beste Lösung	$\overline{x}$ (am Anfang leer)

2. Falls $\mathcal{K} = \emptyset$, dann STOP.

   Ergebnis: Falls $U = -\infty$, so hat (MIP$^=$) keine Lösung.
   Falls $U > -\infty$, so ist U der Optimalwert und $\overline{x}$ eine Optimallösung.

3. **Branching:** Wähle eine Menge $P_j \in \mathcal{K}$.

4. **Bounding:** Löse das durch Weglassen der Ganzzahligkeitsbedingung in P_j entstehende lineare Programm
$$\max\ c^T x,\ x \in LP_j \qquad (\text{LMIP}_j^=)$$

5. Ist $LP_j = \emptyset$ oder (LMIP$_j^=$) unbeschränkt, so sei $c^* = -\infty$, andernfalls sei x^* die Optimallösung von (LMIP$_j^=$) und c^* der Optimalwert.

6. Ist $c^* \leq U$, so entferne P_j aus $\mathcal{K}$.

7. Ist $c^* > U$ und $x_i^* \in \mathbb{Z}\ \forall i \in N_1$, so setze $U := c^*$, $\overline{x} := x^*$ und entferne P_j aus $\mathcal{K}$.

8. **Separation:** Im Falle $c^* > U$ und $x_i^* \notin \mathbb{Z}$ für einige $i \in N_1$ wähle ein $i \in N_1$ mit $x_i \notin \mathbb{Z}$. Entferne P_j aus $\mathcal{K}$ und setze
$$\begin{aligned} P_{k+1} &:= P_j \cap \{x \mid x_i \leq \lfloor x_i^* \rfloor\}, \\ P_{k+2} &:= P_j \cap \{x \mid x_i \geq \lceil x_i^* \rceil\}, \\ k &:= k + 2 \end{aligned}$$
   und gehe zu 2. △

(6.3) Bemerkung. Die Daten des gemischt-ganzzahligen Programms (MIP$^=$) seien – wie in (6.2) verlangt – rational und (LMIP$^=$) sei das durch Weglassen der Ganzzahligkeitsbedingung in (MIP$^=$) entstehende lineare Programm.

(a) Ist (LMIP$^=$) unzulässig oder unbeschränkt, so bricht das Verfahren von Dakin ab und zeigt an, dass (MIP$^=$) keine Lösung oder keine endliche Optimallösung besitzt.

(b) Hat (LMIP$^=$) ein endliches Optimum, und ist $P_0 \neq \emptyset$, so findet das Verfahren von Dakin nach endlich vielen Schritten eine Optimallösung von (MIP$^=$), da alle ganzzahligen Werte der x_i, $i \in N_1$, durchsucht werden.

(c) Hat (LMIP$^=$) ein endliches Optimum, und ist $P_0 = \emptyset$, so kann (theoretisch) durch Einführung einer unteren Schranke für den Zielfunktionswert ein endlicher Abbruch erzwungen werden.

Folglich ist das Dakin-Verfahren ein endliches Verfahren zur Lösung gemischt-ganzzahliger Programme. △

(6.4) Bemerkung. Das ursprüngliche lineare Programm (LP-Relaxierung von (MIP$^=$)) wird bei den verschiedenen Verzweigungsschritten jeweils um sehr einfache Ungleichungen, nämlich um obere bzw. untere Schranken der Variablen erweitert. Für diesen Spezialfall kennen wir bereits eine Variante des Simplexalgorithmus (dualer Simplexalgorithmus mit oberen Schranken, (upper-bounding-technique), siehe ADM I, Kapitel 12), die es erlaubt, die oberen Schranken durch Transformationen im Tableau implizit zu berücksichtigen, ohne die Ungleichungen explizit hinzuzufügen.

Es ist natürlich sinnvoll, zu Beginn des Verfahrens und im Ablauf des Programms eine zulässige Lösung zu finden, um den Wert der unteren Schranke für den Wert der Optimallösung (sukzessive) zu verbessern. △

(6.5) Beispiel. Wir betrachten das folgende rein-ganzzahlige Problem

$$\begin{aligned} \max\ & -7x_1 - 3x_2 - 4x_3 \\ & x_1 + 2x_2 + 3x_3 - x_4 = 8 \\ & 3x_1 + x_2 + x_3 - x_5 = 5 \\ & x_i \geq 0,\ i = 1, \ldots, 5 \\ & x_i \text{ ganzzahlig},\ i = 1, \ldots, 5 \end{aligned}$$

P_0 sei die Lösungsmenge dieses Programms. Eine optimale Lösung des zugehörigen linearen Programms ist gegeben durch $x_3 = x_4 = x_5 = 0$ und

$$x_1 = \frac{2}{5}, \qquad x_2 = \frac{19}{5}, \qquad c^* = -\frac{71}{5}.$$

Wir merken uns bei jedem Programm den Wert der Optimallösung und den gegenwärtigen Wert der unteren Schranke U. Da das Problem rein ganzzahlig ist, muss der Zielfunktionswert rein ganzzahlig sein, wir können also den Wert des LP-Optimums abrunden und erhalten damit eine obere Schranke für alle ganzzahligen Lösungen des gegenwärtig gelösten LP. Für P_0 erhalten wir

$$c^* = -15, \qquad U = -\infty$$

Zur Verzweigung wählen wir die Variable x_2 und fügen einerseits $x_2 \leq 3$ und andererseits $x_2 \geq 4$ zu den Nebenbedingungen von P_0 hinzu. Wir erhalten auf diese Weise neue Kandidatenmengen P_1, P_2. P_0 wird aus $\mathcal{K}$ entfernt. Wir merken uns, was wir getan haben bildlich in dem in Abbildung 6.2 gezeigten Branch & Bound-Baum.

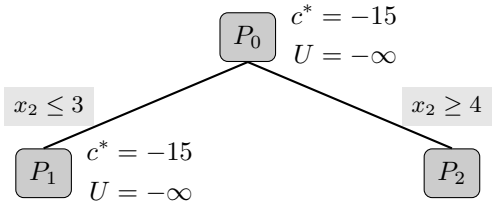


Abbildung 6.2: Erster Separationsschritt im Beispiel (6.5)

Unsere Kandidatenmenge $\mathcal{K}$ enthält nunmehr die Mengen P_1, P_2. Wir entscheiden uns zur Lösung von $\max c^T x$, $x \in P_1$. Die Optimallösung der LP-Relaxierung dieses Problems ist die folgende: $x_4 = x_5 = 0$ und

$$x_1 = \frac{1}{2}, \qquad x_2 = 2, \qquad x_3 = \frac{1}{2}, \qquad c^* = -\frac{29}{2} \text{ (Abrunden ergibt } c^* = -15).$$

Wir tragen dieses Resultat in den Branch & Bound-Baum 6.2 ein. Da die obige Lösung nicht ganzzahlig ist, müssen wir P_1 weiter zerlegen. Wir wählen dazu die Variable x_1, wobei wir zu den Restriktionen von P_1 einerseits $x_1 \geq 1$, andererseits $x_1 \leq 0$ hinzufügen können. Wir entfernen P_1 aus $\mathcal{K}$ und fügen die neuen Kandidatenmengen P_3 und P_4 zu $\mathcal{K}$ hinzu, siehe Abildung 6.3.

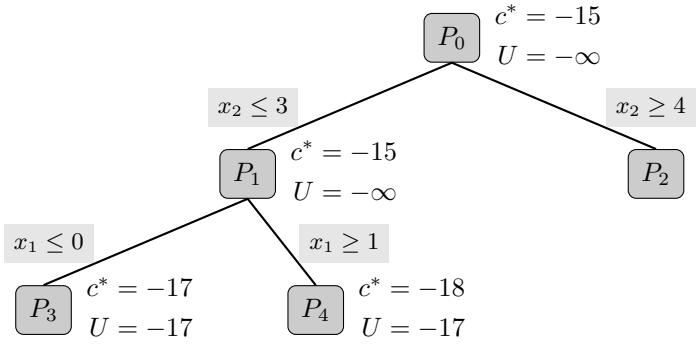


Abbildung 6.3: Zweiter Separationsschritt im Beispiel (6.5)

Die Kandidatenmenge enthält nun P_2, P_3 und P_4. Wir lösen P_3 und erhalten $x_1 = x_5 = 0$ und

$$x_2 = 3, \qquad x_3 = 2, \qquad x_4 = 4, \qquad c^* = -17.$$

Die optimale Lösung der LP-Relaxierung von P_3 ist also ganzzahlig. Wir haben die untere Schranke verbessert und setzen $U := -17$, außerdem ist die Menge P_3 vollständig erledigt, wir können sie aus $\mathcal{K}$ eliminieren.

Es verbleiben P_2 und P_4 in $\mathcal{K}$. Wir entscheiden uns zur Bearbeitung von P_4. Das Optimum der LP-Relaxierung von $\max c^T x$, $x \in P_4$ ist: $x_4 = 0$ und

$$x_1 = 1, \quad x_2 = 3, \quad x_3 = \frac{1}{3}, \quad x_5 = \frac{4}{3}, \quad c^* = -\frac{52}{3} \text{ (Abrunden ergibt } c^* = -18).$$

Also ist der Wert des LP-Optimums kleiner als der Wert der gegenwärtig besten ganzzahligen Lösung. P_4 ist damit auch vollständig erledigt. Die oben erzielten Resultate sind auch in Abbildung 6.3 eingetragen.

In der Kandidatenmenge ist nur noch P_2. Die Optimallösung der LP-Relaxierung von $\max c^T x$, $x \in P_2$ ist $x_3 = x_5 = 0$ und

$$x_1 = \frac{1}{3}, \qquad x_2 = 4, \qquad x_4 = \frac{1}{3}, \qquad c^* = -\frac{43}{3} \text{ (Abrunden ergibt } c^* = -15).$$

Die Lösung ist weder ganzzahlig noch unterschreitet der Optimalwert die gegenwärtig beste Schranke. Wir verzweigen bezüglich x_1 und fügen einerseits $x_1 \leq 0$ und andererseits $x_1 \geq 1$ zu P_2 hinzu. Wir erhalten neue Mengen P_5, P_6, die wir zu $\mathcal{K}$ addieren. P_2 wird aus $\mathcal{K}$ entfernt. Der gegenwärtige Branch & Bound-Baum ist in Abbildung 6.4 gezeigt.

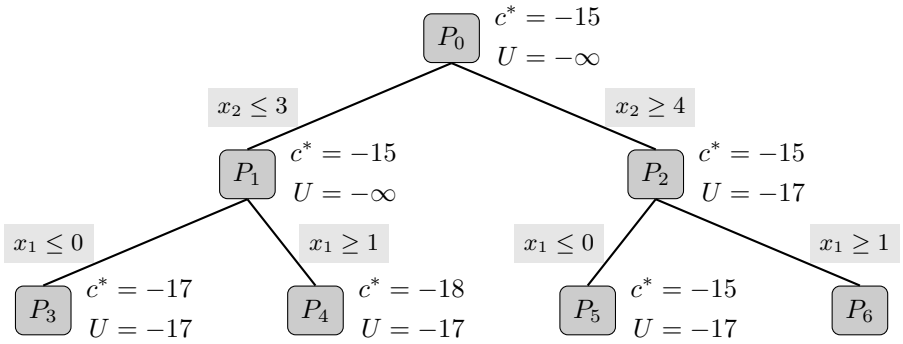


Abbildung 6.4: Dritter Separationsschritt im Beispiel (6.5)

Wir lösen die LP-Relaxierung von $\max c^T x$, $x \in P_5$ und erhalten

$$x_2 = 5, \qquad x_4 = 2, \qquad x_i = 0 \text{ sonst}, \qquad c^* = -15.$$

Diese Lösung ist ganzzahlig, und wir sind mit unserem Algorithmus fertig, da P_6 keine bessere ganzzahlige Lösung enthalten kann, denn eine ganzzahlige Lösung in P_2 hat höchstens den Wert -15. Damit ist eine optimale Lösung des Ausgangsproblems gefunden. △

Ich glaube, dass damit das Prinzip der Branch & Bound-Verfahren klar ist. Wichtig ist, eine Relaxierung zu wählen, die schnell lösbar ist und gute Schranken liefert, also eine gute duale Heuristik auszuwählen. Im Dakin-Verfahren haben wir einfach die kanonische LP-Relaxierung gewählt.

Ein Branch & Bound-Verfahren für das Travelling-Salesman-Problem

Der folgende Teil, in dem wir ein einfaches Beispiel eines Branch & Bound-Verfahrens für das asymmetrische TSP angeben, wird in ADM II nicht mehr behandelt.

Der Großvater aller Branch & Bound-Algorithmen ist das Verfahren von Little, Murty, Sweeny & Karel zur Lösung asymmetrischer Travelling-Salesman-Probleme, das 1963 veröffentlicht wurde und in dem der Name Branch & Bound-Algorithmus geprägt wurde. Weil dieses Verfahren so einfach ist, wollen wir es hier kurz besprechen, jedoch darauf hinweisen, dass es zur Lösung großer Probleme nicht besonders geeignet ist.

(6.6) Algorithmus (Verfahren von Little et al. für das asymmetrische Travelling-Salesman-Problem).

Eingabe: Asymmetrisches TSP auf n Städten gegeben durch Distanzmatrix $c = (c_{ij})$ mit $c_{ii} = \infty$, $i = 1, \ldots, n$ und $c_{ij} \geq 0$.

Ausgabe: Optimallösung des gegebenen TSPs.

Verfahrensweise: Zeilen- und Spaltenreduktion der Matrix C und sukzessive Erzeugung von Teilproblemen. Ein Teilproblem ist definiert durch

$$P(\textsc{Eins}, \textsc{Null}, I, J, C, L, U),$$

wobei
$\textsc{Eins}$ = die Menge der auf 1 fixierten Bögen,
$\textsc{Null}$ = die Menge der auf 0 fixierten Bögen,
C = die Distanzmatrix des Teilproblems,
I = die Zeilenindizes von C,
J = die Spaltenindizes von C,
L = die untere Schranke für das Teilproblem,
U = die obere Schranke für das Globalproblem.

1. Definiere das Anfangsproblem

   $$P(\emptyset, \emptyset, \{1, \ldots, n\}, \{1, \ldots, n\}, C, 0, +\infty),$$

   wobei C die Ausgangsmatrix ist, und setze dieses Problem auf die Problemliste.

2. Sind alle Teilprobleme gelöst $\longrightarrow$ STOP. Andernfalls wähle ein ungelöstes Teilproblem $P(\textsc{Eins}, \textsc{Null}, I, J, C, L, U)$ aus der Problemliste.
   Regel: Wähle das Problem mit kleinster unterer Schranke oder das Problem, bei dem $\textsc{Eins}$ am meisten Elemente enthält. Bei ersterer Wahl erhält man hoffentlich eine optimale Tour, bei zweiterer hat man das Problem schnell abgearbeitet.

3. **Bounding:**

   a) **Zeilenreduktion:**
   FOR all $i \in I$ DO

   Berechne das Minimum der i-ten Zeile von C

   $$c_{ij_0} = \min_{j \in J} c_{ij}$$

   Setze $c_{ij} := c_{ij} - c_{ij_0}$ $\forall j \in J$ und $L := L + c_{ij_0}$.

   END

b) **Spaltenreduktion:**
FOR all $j \in J$ DO

Berechne das Minimum der j-ten Spalte von C

$$c_{i_0 j} = \min_{i \in I} c_{ij}$$

Setze $c_{ij} := c_{ij} - c_{i_0 j}$ $\forall i \in I$ und $L := L + c_{i_0 j}$.

END

c) Ist $L \geq U$, so entferne das gegenwärtige Teilproblem aus der Problemliste und gehe zu 2.

d) Definiere den Nulldigraphen $G_0 = (V, A)$ mit

$$A := \textsc{Eins} \cup \{(i,j) \in I \times J \mid c_{ij} = 0\}.$$

e) Enthält G_0 keine Tour so gehe zu 4.

f) Enthält G_0 eine Tour, so hat die Tour die Länge L.

f_1) Entferne alle Teilprobleme aus der Problemliste, deren untere Schranke größer gleich L ist.

f_2) Setze in allen noch nicht gelösten Teilproblemen $U := L$.

f_3) Gehe zu 2.

(I. A. wird der Nulldigraph G_0 nicht berechnet, sondern so lange enumeriert bis nur noch $(2,2)$-Probleme übriggeblieben sind.)

4. **Branching:**

a) Wähle nach einem Plausibilitätskriterium einen Bogen (i,j), der 0 oder 1 gesetzt wird.
Z. B.: Definiere $u(i,j) := \min\{c_{ik} \mid k \in J \setminus \{j\}\} + \min\{c_{kj} \mid k \in I \setminus \{i\}\} - c_{ij}$, d. h. $u(i,j)$ sind die Zusatzkosten (Umwegkosten), die entstehen, wenn wir von i nach j gehen, ohne (i,j) zu benutzen.
Branching-Regel: Wähle $(i,j) \in I \times J$, so dass

$$c_{ij} = 0 \qquad \text{und} \qquad u(i,j) = \max\{u(p,q) \mid c_{pq} = 0\}.$$

(Wähle also einen Bogen, der zu möglichst hohen Umwegkosten führt.)

b) Definiere die neuen Teilprobleme

b_1) $P(\textsc{Eins} \cup \{(i,j)\}, \textsc{Null}, I \setminus \{i\}, J \setminus \{j\}, C', L, U\}$,
wobei C' aus C durch Streichen der Zeile i und der Spalte j entsteht und $c'_{ji} := \infty$ (zur Vermeidung des Kurzzyklus $\langle i,j \rangle$).

b_2) $P(\textsc{Eins}, \textsc{Null} \cup \{(i,j)\}, I, J, C'', L, U)$,
wobei C'' aus C entsteht durch $c_{ij} := \infty$.

Füge diese Teilprobleme zur Problemliste hinzu und gehe zu 2. △

Das Hauptproblem bei (6.6) ist – wie immer – die Buchhaltung der Teilprobleme. Die Anzahl der Touren in $4.b_1$) ist kleiner als die in $4.b_2$), die Wahl von (i, j) wird so getroffen, dass „hoffentlich“ in $4.b_1$) eine optimale Tour ist.

(6.7) Beispiel. Wir beginnen mit $P = (\emptyset, \emptyset, \{1, \ldots, 6\}, \{1, \ldots, 6\}, C, 0, +\infty\}$

Anfangsmatrix						Zeilenmin ↓							
∞	5	1	2	1	6	1		∞	4	0	1	0	5
6	∞	6	3	7	2	2		4	∞	4	1	5	0
1	4	∞	1	2	5	1	$\longrightarrow$	0	3	∞	0	1	4
4	3	3	∞	5	4	3		1	0	0	∞	2	1
1	5	1	2	∞	5	1		0	4	0	1	∞	4
6	2	6	4	5	∞	2		4	0	4	2	3	∞
						10	Spaltenmin	0	0	0	0	0	0

Die Zeilenreduktion (Schritt 3.a) ergibt insgesamt einen Wert von 10, die Spaltenreduktion erbringt nichts, da nach der Zeilenreduktion bereits jede Spalte eine Null enthält. Die untere Schranke ist somit $L = 10$. Wir können nun den Nulldigraphen G_0 definieren. Er ist in Abbildung 6.5 dargestellt (alle Bögen mit dem Wert 0 sind eingezeichnet.). G_0

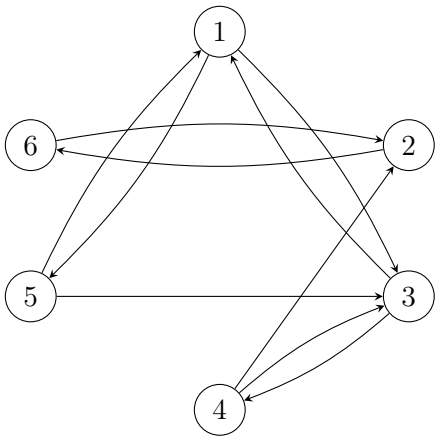


Abbildung 6.5: Nulldigraph G_0

enthält keine Tour.

Wir verwenden die in Schritt 4.a angegebene Branchingregel und wählen den Bogen $(i, j) = (2, 6)$ mit $u(2, 6) = 2$. Das Ausgangsproblem ist nun erledigt. Wir definieren zwei neue Teilprobleme.

Erstes neues Problem: $P = (\{(2,6)\}, \emptyset, \{1,3,4,5,6\}, \{1,2,3,4,5\}, C', 10, \infty)$

		1	*2*	*3*	*4*	*5*	Zeilenmin		*1*	*2*	*3*	*4*	*5*
	1	∞	4	0	1	0	0		∞	4	0	1	0
	3	0	3	∞	0	1	0		0	3	∞	0	1
C' :	*4*	1	0	0	∞	2	0	$\longrightarrow$	1	0	0	∞	2
	5	0	4	0	1	∞	0		0	4	0	1	∞
	6	4	∞	4	2	3	2		2	∞	2	0	1

Die untere Schranke (nach Zeilen- und Spaltenreduktion) für dieses Teilproblem ist $L = 12$.

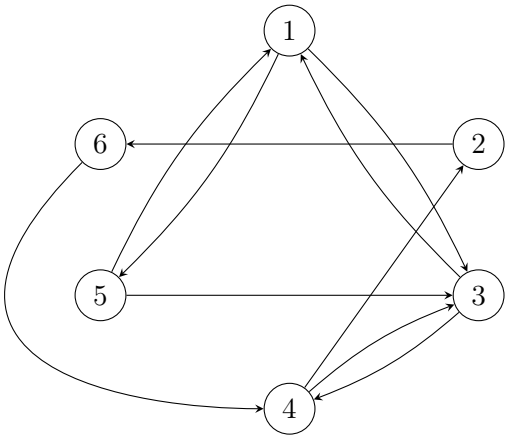


Abbildung 6.6: Nulldigraph nach der obigen Reduktion

Zweites neues Problem: $P = (\emptyset, \{(2,6)\}, \{1,2,3,4,5,6\}, \{1,2,3,4,5,6\}, C'', 10, \infty)$

	∞	4	0	1	0	5	0		∞	4	0	1	0	4
	4	∞	4	1	5	∞	1		3	∞	3	0	4	∞
	0	3	∞	0	1	4	0		0	3	∞	0	1	3
C'' :	1	0	0	∞	2	1	0	$\longrightarrow$	1	0	0	∞	2	0
	0	4	0	1	∞	4	0		0	4	0	1	∞	3
	4	0	4	2	3	∞	0		4	0	4	2	3	∞
	0	0	0	0	0	1	$u = 12$							

△

Die untere Schranke für das zweite Problem ist ebenfalls 12. Das Verfahren wird auf diese Weise fortgeführt. Insgesamt erhält man schließlich den in Abbildung 6.7 dargestellten Branch & Bound-Baum.

Das Verfahren (6.6) ist sehr simpel und (daher) in der Praxis nicht sonderlich effizient. Wesentlich bessere Ergebnisse erzielt man bei Benutzung der Zuordnungsrelaxierung, die wir bisher noch nicht besprochen haben.

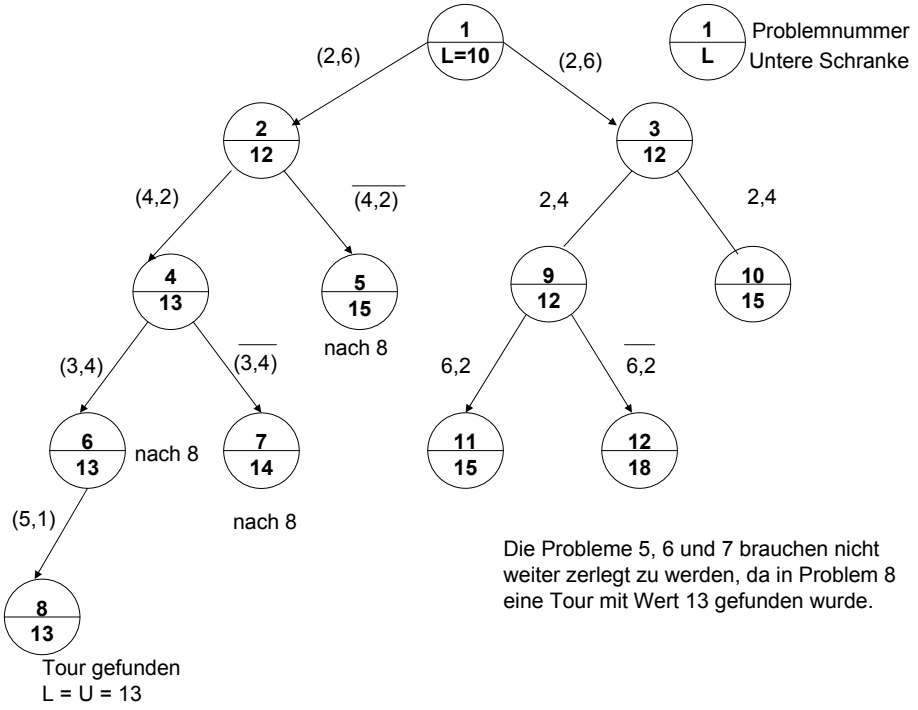


Abbildung 6.7: Branch & Bound-Baum zu Beispiel (6.7)

Der Algorithmus (6.6) erfüllt natürlich nicht die in (6.1) gemachte Annahme, dass das relaxierte Problem „einfach“ zu lösen ist, denn in Schritt 3.e) muss man testen, ob G_0 keine Tour enthält (dieses Problem ist co-$\mathcal{NP}$-schwer), und in Schritt 3.f) wird eine Tour gesucht (was $\mathcal{NP}$-schwer ist). Aber zu der Zeit, als dieser Algorithmus entstand, gab es Komplexitätstheorie noch nicht, und in der Praxis kann man hier natürlich schnelle Heuristiken einsetzen (und muss die Schritte 3.e) und 3.f) ein wenig modifizieren).

7 Theorie der ganzzahligen und gemischt-ganzzahligen Optimierung

Das Ziel des Vorlesungszyklus ADM I – III ist die Bereitstellung von Methoden zur Lösung von linearen, kombinatorischen, ganzzahligen und gemischt-ganzzahligen Optimierungsproblemen. Wir haben sehr ausführlich die verschiedenen Verfahren zur Behandlung linearer Programme vorgestellt, weil diese in den meisten relevanten Fällen die Grundmethodik für alles weitere Vorgehen bilden, insbesondere wenn es um die Lösung von Problemen aus der Praxis geht. Mit dem Branch & Bound-Verfahren haben wir in Kapitel 6 eine theoretisch triviale, in der praktisch-algorithmischen Implementierung jedoch nicht ganz so einfach einzusetzende prinzipielle Methode zur Lösung kombinatorischer, ganzzahliger und gemischt-ganzzahliger Optimierungsprobleme skizziert. Nun soll eine zweite generelle Methode zur Lösung derartiger Probleme vorgestellt werden: das sogenannte Schnittebenenverfahren. Hierzu benötigen wir jedoch ein wenig Theorie, die in den ersten drei Abschnitten dieses Kapitels zu finden ist.

Wir werden uns hauptsächlich auf ganzzahlige Programme konzentrieren und gemischt-ganzzahlige nebenbei (in Anmerkungen) behandeln.

7.1 Einführung

In der Vorlesung haben wir uns bereits intensiv mit der Polyedertheorie beschäftigt. Zur Erinnerung: *Polyeder* sind Teilmengen des $\mathbb{K}^n$ der folgenden Form

$$P(A, b) = \{x \in \mathbb{K}^n \mid Ax \leq b\},$$

d. h. Polyeder sind Durchschnitte von endlich vielen abgeschlossenen Halbräumen. Wir haben gezeigt, dass Polyeder auch eine andere Darstellung besitzen. Es gibt nämlich endliche Mengen $V \subseteq \mathbb{K}^n$ und $E \subseteq \mathbb{K}^n$, so dass gilt

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

Eine *Seitenfläche* eines Polyeders P ist eine Teilmenge $F \subseteq P$ mit der Eigenschaft: Es existiert eine Ungleichung $c^T x \leq c_0$ mit $P \subseteq \{x \in \mathbb{R}^n \mid c^T x \leq c_0\}$ und $F = \{x \in P \mid c^T x = c_0\}$. Wir haben gezeigt, dass zwei Typen von Seitenflächen eine besondere Rolle spielen, nämlich die Ecken und die Facetten, d. h. die minimalen und die maximalen echten Seitenflächen (im Sinne der Mengeninklusion).

Zur Notationsvereinfachung führen wir nun einige neue Bezeichnungen ein.

(7.1) Definition.

(a) *Seien* $A \in \mathbb{K}^{(m,n)}$ *und* $b \in \mathbb{K}^m$*, dann setzen wir*

$$\mathrm{IP}(A,b) := \{x \in \mathbb{Z}^n \mid Ax \leq b\} \qquad \textit{und}$$
$$\mathrm{IP}^{=}(A,b) := \{x \in \mathbb{Z}^n \mid Ax = b, x \geq 0\}.$$

(b) *Seien* $A \in \mathbb{K}^{(m,n_1)}$*,* $B \in \mathbb{K}^{(m,n_2)}$ *und* $b \in \mathbb{K}^m$*, dann setzen wir*

$$\mathrm{MIP}(A,B,b) := \{\tbinom{x}{y} \in \mathbb{K}^{n_1+n_2} \mid Ax + By \leq b, x \in \mathbb{Z}^{n_1}\} \qquad \textit{und}$$
$$\mathrm{MIP}^{=}(A,B,b) := \{\tbinom{x}{y} \in \mathbb{K}^{n_1+n_2} \mid Ax + By = b, x, y \geq 0, x \in \mathbb{Z}^{n_1}\}. \qquad \triangle$$

$\mathrm{IP}(A,b)$ ist also die Menge aller ganzzahligen Vektoren im Polyeder $P(A,b)$ und $\mathrm{IP}^{=}(A,b)$ ist die Menge aller ganzzahligen Vektoren im Polyeder $P^{=}(A,b)$. Analoges gilt für $\mathrm{MIP}(A,B,b)$ und $\mathrm{MIP}^{=}(A,B,b)$. Im Weiteren gehen wir davon aus, dass alle auftretenden ganzzahligen linearen Programme entweder in der Form

$$\max c^T x \quad x \in \mathrm{IP}(A,b) \tag{7.2}$$

oder

$$\max c^T x \quad x \in \mathrm{IP}^{=}(A,b) \tag{7.3}$$

und alle gemischt-ganzzahligen Programme entweder in der Form

$$\max\ c^T x + d^T y \quad (x^T, y^T)^T \in \mathrm{MIP}(A,B,b) \tag{7.4}$$

oder

$$\max\ c^T x + d^T y \quad (x^T, y^T)^T \in \mathrm{MIP}^{=}(A,B,b) \tag{7.5}$$

gegeben sind. Statt des Wortungetüms „ganzzahliges lineares Programm“ schreiben wir in Zukunft häufig *IP* oder *GLP* und statt „gemischt-ganzzahliges lineares Programm“ einfach *MIP* oder *GGLP*. Um die Unterschiede zwischen linearen, ganzzahligen und gemischt-ganzzahligen Programmen und zwischen den Mengen $P(A,b)$, $\mathrm{IP}(A,b)$, $\mathrm{MIP}(A,B,b)$ zu verdeutlichen, betrachten wir zunächst ein Beispiel.

(7.6) Beispiel.

(a) Gegeben sei das folgende IP, dessen Lösungsmenge $\mathrm{IP}(A,b)$ in Abbildung 7.1 dargestellt ist. Die Menge $\mathrm{IP}(A,b)$ enthält 8 Vektoren, die als dicke Punkte gekennzeichnet sind.

$$\begin{aligned} \max\ & 2x_1 + x_2 \\ & x_1 + x_2 \leq 5 \\ & -x_1 + x_2 \leq 0 \\ & 6x_1 + 2x_2 \leq 21 \\ & x_1, x_2 \geq 0 \text{ und ganzzahlig.} \end{aligned}$$

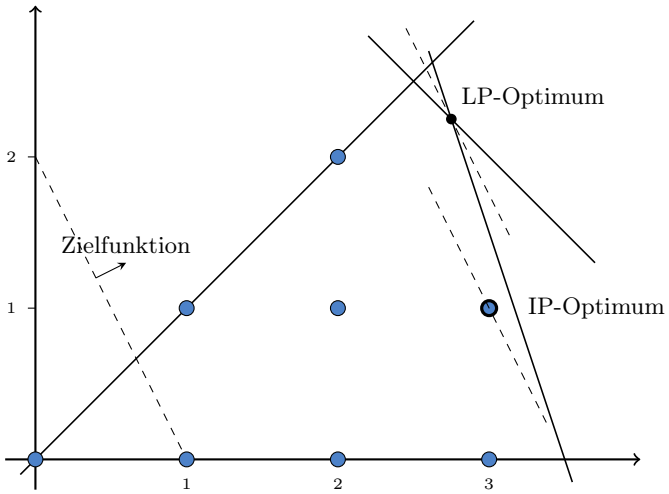


Abbildung 7.1: IP(A, b) in Beispiel (7.6)(a) (farbige Punkte)

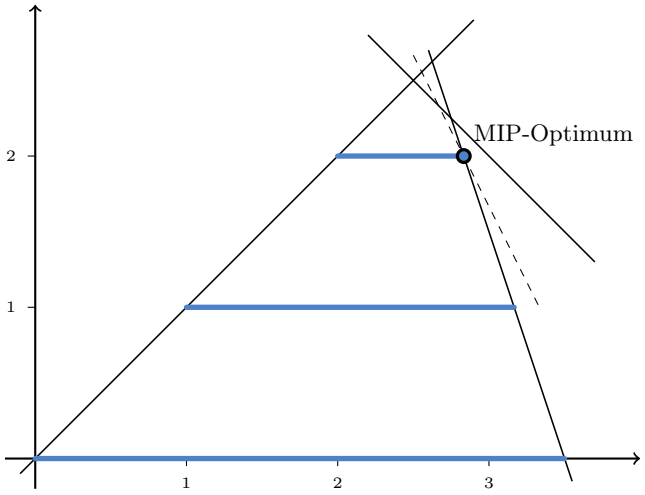


Abbildung 7.2: Lösungsmenge des MIP in Beispiel (7.6)(b) (farbige Streifen)

(b) Wir streichen im obigen IP die Ganzzahligkeitsforderung für x_1 und erhalten so ein MIP. Die Menge der zulässigen Lösungen dieses MIP besteht aus den drei dick gezeichneten Streifen in Abbildung 7.2.

Aus den beiden Abbildungen ist klar, dass die Optimallösungen der LP-Relaxierung des ganzzahligen Programms in (a), des IPs aus (a) und des MIPs aus (b) verschieden sind.△

Das Ziel dieses Kapitels ist die Anwendung polyedertheoretischer Methoden auf IPs und MIPs. Die Mengen IP(A, b), IP$^=(A, b)$, MIP(A, B, b) und MIP$^=(A, B, b)$ sind nach Definition Teilmengen von Polyedern. Wenn man die konvexe Hülle dieser Mengen bildet, so könnte man (intuitiv) meinen, dass dann auch wieder ein Polyeder entsteht. Abbil-

dung 7.3 zeigt z. B. das durch die Ungleichungen aus (7.6)(a) (ohne Ganzzahligkeitsbedingung) definierte Polyeder $P(A, b)$, die konvexe Hülle der Lösungen von (7.6)(b) und die konvexe Hülle der Lösungen von (7.6)(a). Alle diese Mengen sind Polyeder. Wäre dies

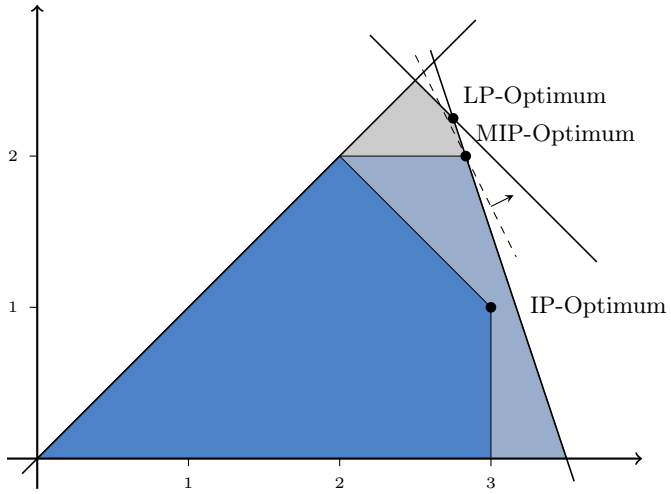


Abbildung 7.3: Konvexe Hüllen der zulässigen Mengen in Beispiel (7.6)(b)

immer so, so könnten wir u. U. alle bereits diskutierten Methoden zur Lösung linearer Programme (Simplexverfahren, Ellipsoidmethode, Innere-Punkte-Verfahren) benutzen, um ganzzahlige Probleme anzugehen. Jedoch ist i. A. die konvexe Hülle von IPs oder MIPs kein Polyeder, wie das nachfolgende Beispiel zeigt.

(7.7) Beispiel. Wir betrachten das Programm

$$\begin{aligned} \max\ & p \\ 0 \le\ & p \le \sqrt{2} \\ & p \in \mathbb{Q}. \end{aligned} \tag{$*$}$$

Bekanntlich ist $\sqrt{2}$ keine rationale Zahl. Jede irrationale Zahl kann aber durch rationale Zahlen beliebig genau (z. B. von unten) angenähert werden. Daher hat $(*)$ kein Maximum, sondern nur ein Supremum, das nicht in einer rationalen Zahl angenommen wird. Wir verwandeln nun $(*)$ in ein ganzzahliges Programm, indem wir die rationale Zahl p als Bruch $\frac{y}{x}$ mit nichtnegativen ganzen Zahlen x, y schreiben. Wir erhalten

$$\begin{aligned} \max\ & -\sqrt{2}x + y \\ & -\sqrt{2}x + y \le 0 \\ & x \ge 1 \\ & y \ge 0 \\ & x, y \text{ ganzzahlig}. \end{aligned} \tag{$**$}$$

Das Programm $(**)$ hat die folgenden Eigenschaften:

(a) Es gibt zulässige ganzzahlige Lösungen.

(b) Der Zielfunktionswert ist nach oben durch 0 beschränkt, da 0 das Optimum des zugehörigen (reellen) LPs ist.

(c) Es gibt keine ganzzahlige Optimallösung (sonst wäre $\sqrt{2}$ rational).

Bezeichnen wir mit S die Lösungsmenge von $(**)$, so folgt aus (a), (b), (c):

$$\text{conv}(S) \text{ ist kein Polyeder.}$$

Denn für jeden Zielfunktionsvektor c gilt trivialerweise $\max\{c^T z \mid z \in S\} = \max\{c^T z \mid z \in \text{conv}(S)\}$. Da für $c^T = (-\sqrt{2}, 1)$ das Problem $\max\{c^T z \mid z \in S\}$ keine Optimallösung hat, hat auch $\max\{c^T z \mid z \in \text{conv}(S)\}$ keine. Wäre $\text{conv}(S)$ ein Polyeder, so müsste, da die Zielfunktion $-\sqrt{2}x + y$ über S und somit über $\text{conv}(S)$ beschränkt ist und da $\text{conv}(S) \neq \emptyset$ gilt, das lineare Programm $\max\{c^T z \mid z \in \text{conv}(S)\}$ eine Optimallösung haben. △

Die Tatsache, dass im obigen Beispiel $\text{conv}(S)$ kein Polyeder ist, hat zwei Gründe:

- Die Daten des Programms sind irrational.
- S ist unbeschränkt.

Wir werden zeigen, dass man unter der Annahme, dass ein Programm eine der beiden Eigenschaften nicht hat, das gewünschte Resultat beweisen kann. Dabei ist die Aussage, dass die konvexe Hülle der ganzzahligen Punkte eines durch rationale Ungleichungen definierten Polyeders ein Polyeder ist, nicht trivial. Sie wird im nächsten Abschnitt behandelt.

(7.8) Satz. *Ist $B \subseteq \mathbb{K}^n$ eine beschränkte Menge, dann ist* $\text{conv}\{x \in B \mid x \text{ ganzzahlig}\}$ *ein Polytop. Ferner gibt es eine ganzzahlige (m,n)-Matrix D und $d \in \mathbb{Z}^m$ mit* $\text{conv}(\{x \in B \mid x \in \mathbb{Z}^n\}) = P(D,d)$. △

Beweis. Ist B beschränkt, so ist $\text{I}B := \{x \in B \mid x \text{ ganzzahlig}\}$ natürlich endlich. Wir wissen, dass die konvexe Hülle einer endlichen Menge ein Polytop ist. Da $\text{I}B \subseteq \mathbb{Q}^n$, ist $\text{conv}(\text{I}B)$ ein Polytop in $\mathbb{Q}^n$. Daraus folgt, dass $\text{conv}(\text{I}B)$ eine Darstellung der Form $P(D', d')$ hat mit D', d' rational. Durch Multiplikation der Zeilen von $D'x \leq d'$ mit geeigneten ganzen Zahlen lässt sich ein System $Dx \leq d$ mit den gewünschten Eigenschaften konstruieren. □

Aus (7.8) folgt unmittelbar, dass für jede Matrix $A \in \mathbb{K}^{(m,n)}$ und jeden Vektor $b \in \mathbb{K}^m$, mit $P(A,b)$ beschränkt, die Menge $\text{conv}(\text{IP}(A,b))$ ein Polytop (mit einer ganzzahligen Darstellung) ist.

7.2 Ganzzahlige Punkte in rationalen Polyedern

Aufgrund der Vorbemerkungen in Abschnitt 7.1 wollen wir uns nun auf Probleme beschränken, bei denen alle Daten rational sind. Zur Abkürzung nennen wir ein Polyeder $P \subseteq \mathbb{K}^n$ *rational*, wenn es eine Matrix $A \in \mathbb{Q}^{(m,n)}$ und einen Vektor $b \in \mathbb{Q}^m$ gibt mit $P = P(A, b)$. Wir wissen, dass ein Polyeder genau dann rational ist, wenn es endliche Mengen $V \subseteq \mathbb{Q}^n$ und $E \subseteq \mathbb{Q}^n$ gibt mit $P = \text{conv}(V) + \text{cone}(E)$. Der Einfachheit halber werden wir uns auch auf rationale Wertebereiche beschränken und nur Polyeder in $\mathbb{Q}^n$ betrachten.

Ist P eine beliebige Teilmenge des reellen oder rationalen Vektorraums, so setzen wir

$$P_{\text{I}} := \text{conv}\{x \in P \mid x \text{ ganzzahlig}\}.$$

Ist also zum Beispiel $P = P(A, b)$ ein Polyeder, so bezeichnet $\text{IP}(A, b)$ die ganzzahligen Punkte in P und P_{I} die konvexe Hülle von $\text{IP}(A, b)$. Wir zeigen nun, dass P_{I} wiederum ein Polyeder ist. Wir wissen dies bereits aus (7.8), falls P beschränkt ist. Ferner gilt offenbar

(7.9) Satz. *Ist C ein rationaler Kegel, so gilt*

$$C = C_{\text{I}}. \qquad \triangle$$

Beweis. Ein rationaler Kegel C hat die Form $C = \text{cone}(E)$, $E \subseteq \mathbb{Q}^n$ endlich. Durch Multiplikation der Vektoren aus E mit geeigneten ganzen Zahlen erhält man eine Menge $E' \subseteq \mathbb{Z}^n$ mit $C = \text{cone}(E')$. Jeder ganzzahlige Vektor in C ist also konische Kombination der Vektoren aus E'. Daraus folgt die Behauptung. □

Damit können wir nun zeigen:

(7.10) Satz. *Ist P ein rationales Polyeder, dann ist P_{I} ebenfalls ein (rationales) Polyeder. Ferner gilt* $\text{rec}(P) = \text{rec}(P_{\text{I}})$. $\triangle$

Beweis. P hat eine Darstellung der Form $P = Q + C$, wobei Q ein Polytop und $C = \text{rec}(P)$ ein Kegel ist. Nach Satz (7.9) können wir annehmen, dass $C = \text{cone}\{y_1, \ldots, y_s\}$ gilt mit $y_i \in \mathbb{Z}^n$, $i = 1, \ldots, s$. Wir setzen

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

und behaupten:

$$P_{\text{I}} = (Q + B)_{\text{I}} + C.$$

Aus dieser Behauptung folgt der Satz, denn $Q + B$ ist beschränkt, also ist $(Q + B)_{\text{I}}$ ein Polytop; und falls $P_{\text{I}} \neq \emptyset$, dann ist C der Rezessionskegel von P_{I}.

Wir beweisen zunächst $P_{\text{I}} \subseteq (Q + B)_{\text{I}} + C$. Es sei p ein ganzzahliger Punkt in P. Dann gibt es Vektoren $q \in Q$ und $c \in C$ mit $p = q + c$. Der Vektor c hat eine Darstellung der

Form $c = \sum_{i=1}^{s} \mu_i y_i$, $\mu_i \geq 0$. Setzen wir $c' := \sum_{i=1}^{s} \lfloor \mu_i \rfloor y_i$ und $b := c - c'$, dann ist c' ganzzahlig; und weil p und c' ganzzahlig sind, ist auch $q + b = p - c'$ ganzzahlig. Ferner gilt nach Definition $b \in B$. Daraus folgt $q + b \in (Q + B)_{\mathrm{I}}$ und somit $p = (q + b) + c' \in (Q + B)_{\mathrm{I}} + C$.

Aus $(Q + B)_{\mathrm{I}} + C \subseteq P_{\mathrm{I}} + C = P_{\mathrm{I}} + C_{\mathrm{I}} = (P + C)_{\mathrm{I}} = P_{\mathrm{I}}$ folgt die umgekehrte Inklusion. □

Analog kann man zeigen:

(7.11) Satz. *Sind $A \in \mathbb{Q}^{(m,n_1)}$, $B \in \mathbb{Q}^{(m,n_2)}$, $b \in \mathbb{Q}^m$, dann ist*

$$\mathrm{conv}\{x \in \mathrm{MIP}(A, B, b)\}$$

ein rationales Polyeder. △

Wir haben schon mehrfach angemerkt, dass für $c \in \mathbb{Q}^n$, $S \subseteq \mathbb{Q}^n$ folgendes gilt:

$$\max\{c^T x \mid x \in S\} = \max\{c^T x \mid x \in \mathrm{conv}(S)\}.$$

Die obigen Sätze zeigen also, dass wir IPs oder MIPs in lineare Programme transformieren können. Haben wir damit eine Lösungsmethode für ganzzahlige und gemischt-ganzzahlige Programme gefunden? Nicht ganz, denn sind IPs oder MIPs durch (7.2), (7.3), (7.4), (7.5) gegeben, so ist überhaupt nicht klar, wie man die Ungleichungen finden kann, die aus den ganzzahligen oder gemischt-ganzzahligen Programmen lineare Programme machen. Wir wissen nun zwar, dass es theoretisch geht, aber ob es praktisch durchführbar ist, ist a priori nicht zu sehen.

Bevor wir dieser Frage nachgehen, wollen wir untersuchen, ob man aus Informationen über die Kodierungslängen der Ungleichungen eines Systems $Ax \leq b$ Abschätzungen über die Kodierungslängen der Facetten von $P(A, b)_{\mathrm{I}}$ ableiten kann. Zunächst beweisen wir einen Hilfssatz.

(7.12) Lemma. *Sei $P \subseteq \mathbb{Q}^n$ ein Polyeder.*

(a) *Hat P eine Darstellung der Form $P = P(A, b)$, so dass die Kodierungslänge jeder Ungleichung des Systems $Ax \leq b$ höchstens φ ist, dann gibt es endliche Mengen $V \subseteq \mathbb{Q}^n$ und $E \subseteq \mathbb{Q}^n$ mit $P = \mathrm{conv}(V) + \mathrm{cone}(E)$, so dass die Vektoren aus $V \cup E$ höchstens die Kodierungslänge $4n^2\varphi$ haben.*

(b) *Hat P eine Darstellung der Form $P = \mathrm{conv}(V) + \mathrm{cone}(E)$, so dass die Kodierungslänge jedes Vektors aus $V \cup E$ höchstens ν ist, dann gibt es ein Ungleichungssystem $Ax \leq b$ mit $P = P(A, b)$, so dass jede Ungleichung des Systems höchstens die Kodierungslänge $3n^2\nu$ hat.* △

Beweis. (a) Angenommen die Voraussetzungen von (a) gelten, dann gibt es Ungleichungen $a_i^T x \leq b_i$, $a_i \in \mathbb{Q}^n$, $b_i \in \mathbb{Q}$, $i = 1, \ldots, m$, die P definieren, mit $\langle a_i \rangle + \langle b_i \rangle \leq \varphi$ für $i = 1, \ldots, m$. Aus der Polyedertheorie wissen wir, dass es endliche Mengen $V, E \subseteq \mathbb{Q}^n$

gibt mit $P = \text{conv}(V) + \text{cone}(E)$, so dass die Komponenten der Vektoren aus $V \cup E$ entweder 0 oder Quotienten von zwei Subdeterminanten der folgenden Matrix sind

$$C = \begin{pmatrix} a_1^T & b_1 \\ \vdots & \vdots \\ a_m^T & b_m \end{pmatrix}.$$

Sei D eine quadratische (k,k)-Untermatrix von C, dann folgt aus einfachen Abschätzungen (Hausaufgabe!)

$$\langle \det D \rangle \leq 2\langle D \rangle - k^2.$$

Da nach Voraussetzung $\langle D \rangle \leq k\varphi$ gilt, erhalten wir

$$\langle \det D \rangle \leq 2\langle D \rangle - k^2 \leq 2n\varphi.$$

Die Zähler und Nenner der Komponenten der Vektoren aus $V \cup E$ haben also höchstens die Kodierungslänge $2n\varphi$, somit hat jede Komponente höchstens die Kodierungslänge $4n\varphi$ und damit jeder dieser Vektoren höchstens die Kodierungslänge $4n^2\varphi$.

(b) beweist man ähnlich. (Hausaufgabe!) □

(7.13) Satz. *Sei $P = P(A,b) \subseteq \mathbb{Q}^n$ ein rationales Polyeder, so dass die Kodierungslänge jeder Ungleichung des Systems $Ax \leq b$ höchstens φ ist. Dann gibt es ein Ungleichungssystem $Dx \leq d$ mit $P_I = P(D,d)$, so dass die Kodierungslänge jeder Ungleichung des Systems höchstens $15n^6\varphi$ ist.* △

Beweis. Ist $P_I = \emptyset$, so ist die Behauptung trivial. Wir können also annehmen, dass $P_I \neq \emptyset$ gilt.

Nach Lemma (7.12) gibt es eine Darstellung von P der Form

$$P = \text{conv}\{v_1, \ldots, v_t\} + \text{cone}\{y_1, \ldots, y_s\},$$

so dass jeder der Vektoren $v_1, \ldots, v_t$, $y_1, \ldots, y_s$ höchstens die Kodierungslänge $4n^2\varphi$ hat. Wir setzen

$$Q := P \cap \{x \in \mathbb{Q}^n \mid -(n+1) \cdot 2^{4n^3\varphi} \leq x_i \leq (n+1) \cdot 2^{4n^3\varphi}, i = 1, \ldots, n\} \qquad (7.14)$$

und zeigen

$$P_I = Q_I + \text{cone}\{y_1, \ldots, y_s\} \qquad (*)$$

Die Inklusion $\supseteq$ in $(*)$ gilt trivialerweise. Um die umgekehrte Inklusion zu zeigen, wählen wir einen beliebigen (ganzzahligen) Vektor $x \in P$. Für $j = 1, \ldots, s$ sei y_j' derjenige ganzzahlige Vektor, der aus y_j dadurch entsteht, dass man y_j mit dem kgV der Nenner der Komponenten von y_j multipliziert. Die Kodierungslänge von y_j' ist somit höchstens $4n^3\varphi$. Aufgrund des Satzes von Caratheodory (1.61) können wir x in folgender Form darstellen

$$x = \lambda_1 v_1 + \cdots + \lambda_t v_t + \mu_1 y_1' + \cdots + \mu_s y_s',$$

wobei $\lambda_1, \ldots, \lambda_t, \mu_1, \ldots, \mu_s \geq 0$, $\lambda_1 + \cdots + \lambda_t = 1$ und höchstens n der rationalen Zahlen μ_j von Null verschieden sind. Wir setzen nun

$$x' := \lambda_1 v_1 + \cdots + \lambda_t v_t + (\mu_1 - \lfloor \mu_1 \rfloor) y'_1 + \cdots + (\mu_s - \lfloor \mu_s \rfloor) y'_s$$
$$x'' := \lfloor \mu_1 \rfloor y'_1 + \ldots + \lfloor \mu_s \rfloor y'_s$$

Da x und x'' ganzzahlig sind, ist auch $x' = x - x''$ ganzzahlig. Für jeden der Vektoren v_i und y'_j hat jede seiner Komponenten einen Betrag, der nicht größer als $2^{4n^3\varphi}$ ist. Da höchstens n der Zahlen $\mu_1, \ldots, \mu_s$ von Null verschieden sind, gilt $\lambda_1 + \cdots + \lambda_t + (\mu_1 - \lfloor \mu_1 \rfloor) + \cdots + (\mu_s - \lfloor \mu_s \rfloor) \leq n + 1$; also hat jede Komponente von x' einen Absolutbetrag, der nicht größer als $(n+1)2^{4n^3\varphi}$ ist. Mithin gilt $x' \in Q$. Da $x'' \in \text{cone}\{y'_1, \ldots, y'_s\}$, gehört $x = x' + x''$ zu $Q_\text{I} + \text{cone}\{y_1, \ldots, y_s\}$. Daraus folgt die Gleichheit in $(*)$.

Sei Z die Menge der ganzzahligen Vektoren in Q. Nach Definition hat jeder Vektor in Z höchstens die Kodierungslänge $5n^4\varphi$. Aufgrund von $(*)$ gilt

$$P_\text{I} = \text{conv}(Z) + \text{cone}\{y_1, \ldots, y_s\}.$$

In dieser Darstellung von P_I als Summe eines Polytops und eines Kegels hat jeder Vektor höchstens die Kodierungslänge $5n^4\varphi$; folglich können wir aus Lemma (7.12)(b) schließen, dass P_I durch ein Ungleichungssystem $Dx \leq d$ dargestellt werden kann, so dass jede Ungleichung dieses Systems höchstens die Kodierungslänge $15n^6\varphi$ hat. □

Im obigen Beweis haben wir eine Aussage mitbewiesen, die für weitere Anwendungen interessant ist.

(7.15) Korollar. *Sei $P = P(A, b)$ ein rationales Polyeder, so dass die Kodierungslänge jeder Ungleichung des Systems $Ax \leq b$ höchstens φ ist. Falls P einen ganzzahligen Vektor enthält, dann gibt es in P einen ganzzahligen Vektor, der im Würfel*

$$\{x \in \mathbb{Q}^n \mid -(n+1) \cdot 2^{4n^3\varphi} \leq x_i \leq (n+1) \cdot 2^{4n^3\varphi}\}$$

enthalten ist, und es gibt in P einen ganzzahligen Vektor der Kodierungslänge höchstens $5n^4\varphi$. △

Aus dieser Folgerung ergibt sich eine wichtige komplexitätstheoretische Konsequenz.

(7.16) Korollar. *Das Entscheidungsproblem „Hat ein gegebenes rationales Ungleichungssystem $Ax \leq b$ eine ganzzahlige Lösung?“ ist in der Klasse $\mathcal{NP}$.* △

Beweis. Um (nichtdeterministisch) zu zeigen, dass eine ganzzahlige Lösung von $Ax \leq b$ existiert, können wir einfach eine Lösung y raten. Die Korrektheit der geratenen Lösung y kann aber nur dann (durch Substitution von y in das Ungleichungssystem) in polynomialer Zeit überprüft werden, wenn die Kodierungslänge von y polynomial in $\langle A \rangle + \langle b \rangle$ ist. Aus (7.15) folgt, dass – wenn $Ax \leq b$ überhaupt eine ganzzahlige Lösung hat – eine ganzzahlige Lösung y mit $\langle y \rangle \leq 5n^4 \langle A, b \rangle$ existiert. Also kann man tatsächlich eine „kleine“ ganzzahlige Lösung von $Ax \leq b$ raten, wenn das Entscheidungsproblem eine Ja-Antwort besitzt. □

Ein weiteres interessantes Korollar von (7.13) ist die folgende Beobachtung.

(7.17) Korollar. *Sei $P \subseteq \mathbb{Q}^n$ ein Polyeder, für das es eine Beschreibung $P = P(A, b)$ gibt, so dass jede der Ungleichungen des Systems höchstens die Kodierungslänge φ hat, und sei $c \in \mathbb{Q}^n$. Falls $\max\{c^T x \mid x \in P, x \text{ ganzzahlig}\}$ endlich ist, dann gibt es eine Optimallösung, deren Kodierungslänge höchstens $5n^4\varphi$ ist, und der Absolutbetrag des Optimalwertes dieses IPs ist höchstens $2^{\langle c\rangle+5n^4\varphi-2n}$.* △

Beweis. Aus dem Beweis von Satz (7.13) folgt, dass das Maximum von $\max\{c^T x \mid x \in P, x \text{ ganzzahlig}\}$ in einem der Vektoren $z \in Z$ (das sind die ganzzahligen Vektoren aus Q in (7.14)) angenommen wird. Aus $\langle z\rangle \leq 5n^4\varphi$ folgt die erste Behauptung. Die zweite folgt mit Cauchy-Schwarz und Lemma (4.6)(c) aus

$$|c^T x| \;\leq\; \|c\|_2\|z\|_2 \;\leq\; (2^{\langle c\rangle-n}-1)(2^{\langle z\rangle-n}-1) \;\leq\; 2^{\langle c\rangle+5n^4\varphi-2n}. \qquad \square$$

Folgerung (7.17) bzw. (7.15) impliziert, dass für ein ganzzahliges Programm $\max c^T x$, $x \in \mathrm{IP}(A, b)$ mit $c \in \mathbb{Z}^n$ die Menge der möglichen Optimallösungen im Würfel

$$\{x \in \mathbb{Z}^n \mid -(n+1)\cdot 2^{4n^3\langle A,b\rangle} \leq x_i \leq (n+1)\cdot 2^{4n^3\langle A,b\rangle}\}$$

liegt und dass der Optimalwert dieses IPs, wenn er existiert, eine ganze Zahl z^* im folgenden Intervall ist

$$-2^{\langle c\rangle+5n^4\langle A,b\rangle-2n} \leq z^* \leq 2^{\langle c\rangle+5n^4\langle A,b\rangle-2n}.$$

Daraus folgt direkt, dass IPs durch binäre Suche gelöst werden können, wir haben dazu „lediglich“ die Entscheidungsfrage

„Gibt es einen Vektor $x \in \mathrm{IP}(A, b)$ mit $c^T x \geq z^*$?“

für die z^* aus dem obigen Intervall zu lösen. Leider ist jedoch das letztere Entscheidungsproblem $\mathcal{NP}$-vollständig.

Analoge Resultate wie die oben für ganzzahlige Programme bewiesenen, lassen sich auch für gemischt-ganzzahlige Programme ableiten. Aus ihnen folgt die Endlichkeit des Dakin-Verfahrens (6.2).

7.3 Schnittebenentheorie

Wir haben im vorhergehenden Abschnitt gezeigt, dass für jedes rationale Polyeder P die Menge P_{I}, die konvexe Hülle der ganzzahligen Punkte in P, ein Polyeder ist (P_{I} nennt man häufig das *zu P gehörige ganzzahlige Polyeder*). Ferner haben wir bewiesen, dass es für $P = P(A, b)$ eine Darstellung $P_{\mathrm{I}} = P(D, d)$ gibt, so dass die Ungleichungen des Systems $Dx \leq d$ „kleine“ Koeffizienten haben. Die Frage stellt sich nun, ob man $Dx \leq d$ algorithmisch bestimmen kann und ob man die Anzahl der Ungleichungen von $Dx \leq d$ durch ein Polynom in der Anzahl der Ungleichungen von $Ax \leq b$ beschränken kann.

Die erste Frage hat eine positive Antwort, die zweite nicht. Selbst dann, wenn ein kombinatorisches Optimierungsproblem in polynomialer Zeit gelöst werden kann, kann die Anzahl der Ungleichungen, die notwendig sind, um die konvexe Hülle der zulässigen Punkte zu beschreiben, exponentiell in den Daten des Problems sein. Wir haben hierzu bereits mehrere Beispiele in Kapitel 3 gegeben. Wir erinnern kurz an das Matching-Problem, das wir ausführlich in Abschnitt 3.3 behandelt haben.

Ein *Matching* M ist eine Teilmenge der Kantenmenge eines Graphen $G = (V, E)$, so dass jeder Knoten $v \in V$ auf höchstens einer Kante von M liegt.

Das *Matching-Problem* ist die Aufgabe, in einem Graphen $G = (V, E)$ mit Kantengewichten $c_e \in \mathbb{Q}$ für alle $e \in E$ ein Matching maximalen Gewichts zu finden. Aus der Definition können wir eine Formulierung des Matching-Problems als ganzzahliges Programm ableiten. Diese lautet

$$\begin{aligned} &\max\ c^T x \\ x(\delta(v)) := \sum_{e \in \delta(v)} x_e &\le 1 \qquad \forall v \in V \\ x_e &\ge 0 \qquad \forall e \in E \\ x_e &\in \{0,1\} \quad \forall e \in E. \end{aligned} \tag{7.18}$$

Hierbei ist $\delta(v)$ die Menge aller Kanten, die den Knoten v enthalten. Bezeichnen wir mit $P(G)$ die Menge der zulässigen Lösungen von (7.18) ohne Ganzzahligkeitsbedingung und mit $\mathrm{MATCH}(G)$ die konvexe Hülle aller Inzidenzvektoren von Matchings in G, so gilt offenbar

$$P(G)_{\mathrm{I}} = \mathrm{MATCH}(G).$$

Man kann zeigen, siehe Satz (3.14), dass

$$P(G) = \mathrm{MATCH}(G)$$

genau dann gilt, wenn G bipartit ist und keine isolierten Knoten hat. Man kann ferner zeigen (siehe Satz (3.18), Edmonds (1965)), dass $\mathrm{MATCH}(G)$ durch das folgende System von Ungleichungen beschrieben werden kann:

$$\begin{aligned} x(\delta(v)) &\le 1 \qquad\qquad \forall v \in V \\ x(E(W)) &\le \frac{1}{2}(|W| - 1) \quad \forall W \subseteq V, |W| \ge 3 \text{ und ungerade} \\ x_e &\ge 0 \qquad\qquad \forall e \in E. \end{aligned} \tag{7.19}$$

Hierbei bezeichnet $E(W)$ die Menge aller Kanten, bei denen beide Endknoten in W liegen.

Falls G der vollständige Graph K_n, $n \ge 3$ ist, ist das System (7.19) nicht redundant, d. h. jede der in (7.19) angegebenen Ungleichungen definiert eine Facette von $\mathrm{MATCH}(K_n)$. Somit hat $\mathrm{MATCH}(K_n)$ rund 2^{n-1} Facetten; daraus folgt, dass die Speicherkomplexität jeder vollständigen Beschreibung von $\mathrm{MATCH}(K_n)$ exponentiell in der Kodierungslänge von K_n bzw. (7.18) ist. Folglich gibt es (für das Matching-Polyeder und

somit generell) keinen Algorithmus, der aus einem Ungleichungssystem $Ax \leq b$ eine vollständige Beschreibung von $P(A,b)_{\mathrm{I}}$ der Form $P(A,b)_{\mathrm{I}} = \{x \mid Dx \leq d\}$ konstruiert und dessen Laufzeit polynomial in $\langle A\rangle + \langle b\rangle$ ist.

Trotz dieses negativen Resultates ist es – wie wir noch sehen werden – sinnvoll, sich Gedanken zu machen, wie man aus $Ax \leq b$ ein System $Dx \leq d$ mit $P(A,b)_{\mathrm{I}} = P(D,d)$ algorithmisch konstruieren kann. Dies wird mit sogenannten Schnittebenenmethoden gemacht. Hier wollen wir den theoretischen Hintergrund dieser Verfahren erläutern.

(7.20) Definition. *Es seien $A \in \mathbb{Q}^{(m,n)}$ und $b \in \mathbb{Q}^m$. Mit S bezeichnen wir das System der linearen Ungleichungen $Ax \leq b$, und wir setzen $P := P(A,b)$.*

(a) *Wir sagen, dass eine Ungleichung $d^T x \leq d_0$, $d \in \mathbb{Z}^n$, zum* elementaren Abschluss *von S gehört, wenn es einen Vektor $\lambda \in \mathbb{Q}^m$, $\lambda \geq 0$ gibt mit*

$$\lambda^T A = d^T \qquad \text{und}$$
$$\lfloor \lambda^T b \rfloor \leq d_0.$$

(b) *Wir setzen $e^0(S) := S$ und bezeichnen die Menge aller Ungleichungen, die zum elementaren Abschluss von S gehören mit $e^1(S)$.*

(c) *Für $k > 1$ definieren wir $e^k(S) := e^1(S \cup e^{k-1}(S))$.*

(d) $\mathrm{cl}(S) := \bigcup_{k=1}^{\infty} e^k(S)$ *heißt der* Abschluss *von S.*

(e) *Wir setzen $P^0 := P$ und*

$$P^k := \{x \in \mathbb{Q}^n \mid x \text{ erfüllt alle Ungleichungen des Systems } e^k(S)\}. \qquad \triangle$$

Der (einfache) Hintergrund der obigen Begriffsbildung ist der folgende. Für jeden Vektor $\lambda \in \mathbb{Q}^m$, $\lambda \geq 0$ wird die Ungleichung

$$(\lambda^T A)x \leq \lambda^T b$$

von allen Punkten aus $P(A,b)$ erfüllt. Ist $\lambda^T A$ ein ganzzahliger Vektor, so ist für jeden Vektor $x \in \mathrm{IP}(A,b)$ der Wert $(\lambda^T A)x$ eine ganze Zahl. Folglich erfüllt in diesem Falle jeder Vektor $x \in P(A,b)_{\mathrm{I}}$ die Ungleichung

$$(\lambda^T A)x \leq \lfloor \lambda^T b \rfloor.$$

Diese Ungleichung (also eine Ungleichung des elementaren Abschlusses) ist somit gültig für das Polyeder $P(A,b)_{\mathrm{I}}$. Durch Induktion folgt, dass jede Ungleichung aus dem Abschluss $\mathrm{cl}(S)$ von S gültig bezüglich $P(A,b)_{\mathrm{I}}$ ist. Es ist relativ erstaunlich, dass $\mathrm{cl}(S)$ alle Ungleichungen enthält, die Facetten von $P(A,b)_{\mathrm{I}}$ definieren. Der folgende Satz fasst die Eigenschaften der Abschlussoperationen zusammen.

(7.21) Satz. *$P = P(A,b) \subseteq \mathbb{Q}^n$ sei ein rationales Polyeder, dann gilt:*

(a) *Ist* $c^T x \le c_0$ *mit* $c \in \mathbb{Z}^n$, $c_0 \in \mathbb{Z}$ *gültig bezüglich* P_I, *dann ist* $c^T x \le c_0$ *ein Element von* $\mathrm{cl}(Ax \le b)$.

(b) *Es gibt eine ganze Zahl* $k \ge 0$, *so dass* $\mathrm{cl}(Ax \le b) = e^k(Ax \le b)$ *gilt und endlich viele der Ungleichungen aus* $e^k(Ax \le b)$ *das Polyeder* P_I *definieren, d. h. es gibt ein* $k \ge 0$ *mit*

$$P_\mathrm{I} = P^k.$$

(c) *Sei* $c \in \mathbb{Z}^n$, *dann gibt es eine ganze Zahl* $k \ge 0$, *so dass für das ganzzahlige Programm*

$$\max\{c^T x \mid x \in \mathrm{IP}(A, b)\} \qquad (*)$$

und das lineare Programm

$$\max\{c^T x \mid x \in P^k\} \qquad (**)$$

gilt: $(*)$ *hat keine bzw. keine endliche Lösung genau dann, wenn* $(**)$ *keine bzw. keine endlichen hat. Sind* $(*)$ *und* $(**)$ *endlich lösbar, so stimmen die Maxima überein.* △

Beweis. Siehe Chvátal (1973) für beschränkte Polyeder und Schrijver (1980) für den allgemeinen Fall. □

Man kann anhand von Beispielen zeigen, dass das k in Satz (7.21)(b) bzw. (c) beliebig groß sein kann, siehe Chvátal (1973). Mit einigem Recht kann man sagen, dass ein Problem mit „großem k“ schwieriger ist als eines mit „kleinem“. Die Zahl k misst in einem gewissen Sinne die „Nähe zur Ganzzahligkeit“ eines linearen Programmierungsproblems.

Trotzdem bleibt festzuhalten, dass man nach Satz (7.21) aus den Daten eines Polytops $P(A, b)$ prinzipiell das definierende Ungleichungssystem des zugehörigen ganzzahligen Polytops $P_\mathrm{I} = \mathrm{conv}(\mathrm{IP}(A, b))$ konstruieren kann. Wir werden im nächsten Kapitel sehen, dass dieses Verfahren sogar algorithmisch ausgewertet werden kann und zu einem endlichen Algorithmus führt. Zum Abschluss dieses Abschnitts betrachten wir noch ein Beispiel.

(7.22) Beispiel.

(a) Gegeben sei das folgende Ungleichungssystem $Ax \le b$

$$\begin{aligned} x_1 + 2x_2 &\le 3 && (1)\\ x_1 &\le 1 && (2)\\ -x_1 &\le 0 && (3)\\ -x_2 &\le 0 && (4) \end{aligned}$$

dessen Lösungsmenge $P(A, b)$ in Abbildung 7.4 dargestellt ist. Es gilt $\mathrm{IP}(A, b) = \{\binom{0}{0}, \binom{0}{1}, \binom{1}{0}, \binom{1}{1}\}$ und damit

$$P_\mathrm{I} = \mathrm{conv}(\mathrm{IP}(A, b)) = \{x \in \mathbb{R}^2 \mid x_1 \le 1, x_2 \le 1, x_1 \ge 0, x_2 \ge 0\}.$$

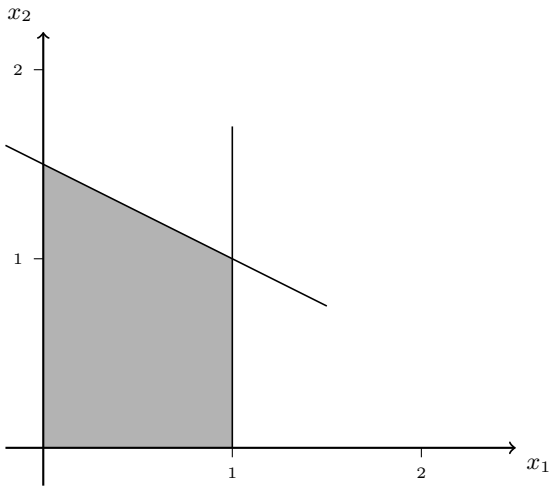


Abbildung 7.4: Lösungsmenge des Ungleichungssystems aus Beispiel (7.22)(a)

Zum Beispiel gilt $x_1 + x_2 \leq 2 \in e^1(S)$, denn

$$(1) + (2) \implies 2x_1 + 2x_2 \leq 4 \implies x_1 + x_2 \leq 2.$$

Zur Beschreibung von $\text{conv}(\text{IP}(A, b))$ fehlt noch die Ungleichung $x_2 \leq 1$, aber auch diese ist in $e^1(S)$ enthalten, denn

$$(1) + (3) \implies 2x_2 \leq 3 \implies x_2 \leq \frac{3}{2} \implies x_2 \leq 1 = \left\lfloor \frac{3}{2} \right\rfloor.$$

Damit haben wir gezeigt $\text{cl}(S) = e^1(S)$, d. h.

$$P_{\text{I}} = \text{conv}(\text{IP}(A, b)) = \bigcap \{x \in \mathbb{R}^2 \mid a^T x \leq a_0 \in e^1(S)\}.$$

(b) Hausaufgabe!
Gegeben sei das folgende Ungleichungssystem S

$$-kx + y \leq 0 \tag{1}$$
$$kx + y \leq k \tag{2}$$
$$-y \leq 0 \tag{3}$$

dessen Lösungsmenge in Abbildung 7.5 zu sehen ist. Die ganzzahligen Lösungen von S sind die Vektoren $\binom{0}{0}, \binom{1}{0}$. Zeigen Sie

i) Die Ungleichungen

$$y \leq \left\lfloor \tfrac{k}{2} \right\rfloor \tag{4}$$
$$-y \leq 0 \tag{5}$$
$$x \leq 1 \tag{6}$$

gehören zu $e^1(S)$.

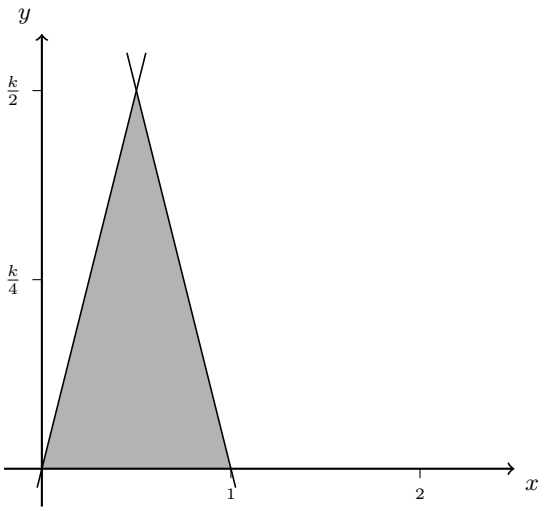


Abbildung 7.5: Lösungsmenge des Ungleichungssystems aus Beispiel (7.22)(b)

ii) Die Ungleichung

$$y \leq 0 \tag{7}$$

gehört nicht zu $e^1(S)$, falls $k \geq 2$ (offenbar gehört (7) zu $\mathrm{cl}(S)$). △

(7.23) Bemerkung (Hausaufgabe). Zeigen Sie, dass für jeden Graphen G gilt

$$\mathrm{MATCH}(G) = P(G)^1,$$

d. h. das Ungleichungssystem (7.19) gehört zum elementaren Abschluss des Ungleichungssystems (7.18). △

7.4 Ein Schnittebenenverfahren für ganzzahlige Programme

In Abschnitt 7.1 haben wir gesehen, dass die konvexe Hülle der Lösungen eines ganzzahligen oder gemischt-ganzzahligen Programms nicht unbedingt ein Polyeder sein muss (Beispiel (7.7)). Unter der Voraussetzung der Rationalität der Daten oder Beschränktheit der Lösungsmenge kann jedoch gezeigt werden (Sätze (7.8), (7.10)), dass diese konvexe Hülle immer ein Polyeder ist. Da wir zur Lösung ganzzahliger und gemischt-ganzzahliger Programme das Simplexverfahren bzw. andere Verfahren der linearen Programmierung einsetzen wollen, werden wir voraussetzen müssen, dass alle Daten rational sind. Für Probleme aus der Praxis ist dies immer der Fall, da auf einem Rechner sowieso nur rationale Zahlen repräsentiert werden können. Haben wir ein Ungleichungssystem $Ax \leq b$ mit rationalen Daten, so kann jedes Matrixelement als Bruch mit positivem Nenner geschrieben werden, das gleiche gilt für den Vektor b. Daher können wir für jede Ungleichung $A_{i.}x \leq b_i$ das kleinste gemeinsame Vielfache k_i der Nenner bestimmen. Die Ungleichung

$$k_i A_{i.} \leq k_i b_i$$

definiert denselben Halbraum und hat nur ganzzahlige Elemente. Auf diese Weise erhalten wir ein ganzzahliges System $Dx \leq d$, so dass $P(A,b) = P(D,d)$ gilt. Daraus folgt, dass es keine Einschränkung der Allgemeinheit ist, wenn wir statt Rationalität der Daten Ganzzahligkeit der Daten fordern.

> **Im weiteren Verlauf dieses Kapitels seien alle Daten von $Ax \leq b$, bzw. $Ax = b$, etc. ganzzahlig.**

Wir wissen also, dass bei ganzzahligen Daten die konvexe Hülle der Lösungen eines IP oder MIP ein Polyeder ist und dass der Optimalwert eines ganzzahligen oder gemischt-ganzzahligen Programms mit dem Optimalwert des linearen Programms über der konvexen Hülle übereinstimmt. Ferner haben wir in Satz (7.21) gesehen, dass ein lineares System zur Beschreibung der konvexen Hülle in einem gewissen Sinne „effektiv“ konstruierbar ist. Wir werden uns nun damit beschäftigen, wie diese polyedertheoretischen Erkenntnisse mit Hilfe von Schnittebenenverfahren algorithmisch ausgewertet werden können.

Schnittebenenverfahren für IPs arbeiten im Prinzip wie folgt:

(7.24) Algorithmus (Allgemeines Schnittebenenverfahren für ganzzahlige Programme).
Eingabe: $A \in \mathbb{Z}^{(m,n)}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$.
Ausgabe: Optimale Lösung des ganzzahligen Programms

$$\begin{aligned} \max\ & c^T x \\ & Ax = b \\ & x \geq 0 \\ & x \text{ ganzzahlig.} \end{aligned} \tag{IP}$$

1. Löse das zu (IP) gehörige lineare Programm (LP), das durch Weglassen der Ganzzahligkeitsbedingung entsteht. Hat (LP) keine endliche Optimallösung, so ist (IP) unbeschränkt, bzw. $\mathrm{IP}^{=}(A,b) = \emptyset$. Falls (LP) beschränkt ist, so sei x^* eine Optimallösung von (LP).

2. Ist x^* ganzzahlig $\longrightarrow$ STOP (gib x^* aus).
   Andernfalls bestimme eine Ungleichung $d^T x \leq d_0$ mit folgenden Eigenschaften (*Schnittebene*):

$$\begin{aligned} d^T x^* &> d_0 \\ d^T x &\leq d_0 \qquad \forall x \in \mathrm{IP}^{=}(A,b) \end{aligned}$$

3. Füge die Ungleichung $d^T x \leq d_0$ unter Hinzufügung einer Schlupfvariablen zum gegenwärtigen System hinzu. Nenne dieses neue System (IP) und gehe zu 1. $\triangle$

Für ganzzahlige Programme der Form (7.2) und für gemischt-ganzzahlige Programme verlaufen Schnittebenenverfahren analog.

Schnittebenenverfahren unterscheiden sich nur durch die Art der Bestimmung der Schnittebenen – dafür gibt es verschiedene Techniken – und durch die Art der Lösung des erweiterten linearen Programms. Im Allgemeinen wird Schritt 1 bei der ersten Ausführung mit dem primalen Simplexverfahren gelöst und nach jeder Ausführung von Schritt 3 mit Hilfe der dualen Simplexmethode.

Der Name Schnittebene kommt daher, dass die gegenwärtige Optimallösung abgeschnitten wird, siehe Abbildung 7.6.

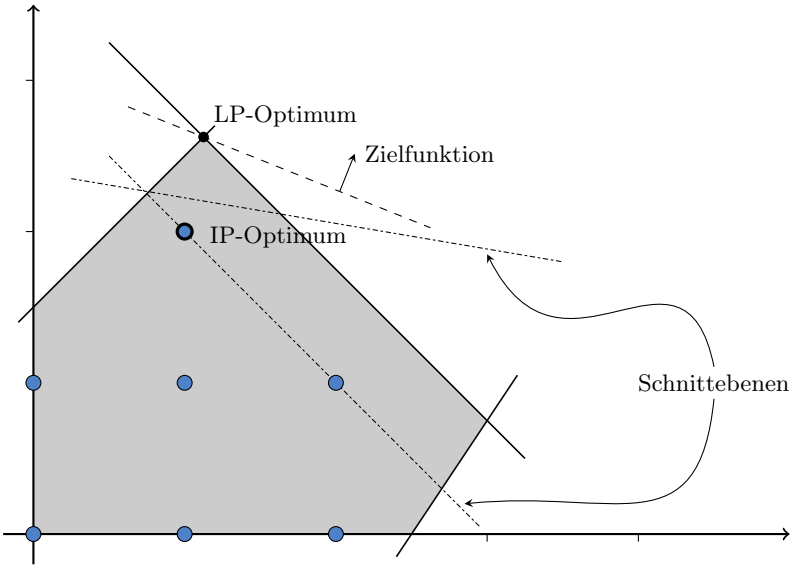


Abbildung 7.6: Schnittebenen schneiden nicht-ganzzahlige Optimallösungen ab

Satz (7.21) besagt, dass mit Hilfe des Abschlussoperators cl im Prinzip Schnittebenen konstruiert werden können, so dass irgendwann die konvexe Hülle der ganzzahligen Punkte erreicht ist. Die Frage ist nun, ob eine (möglichst einfache) Regel angegeben werden kann, die Schnittebenen erzeugt und endliche Konvergenz garantiert. Bis zum Jahre 1958 sind verschiedene Regeln konstruiert worden, die aber alle nicht funktionierten. Gomory gelang dann 1958 die Entwicklung eines endlichen Schnittebenenverfahrens für (rein) ganzzahlige Programme. Später hat er weitere Techniken zur Schnittebenenerzeugung entworfen, die auch im Falle gemischt-ganzzahliger Programme arbeiten.

Bevor Gomory seine Methode veröffentlichte, war von Dantzig (siehe (Garfinkel und Nemhauser, 1972, S. 166)) der folgende Schnitt vorgeschlagen worden. Ist eine optimale Lösung des zu (7.3) gehörigen LP nicht ganzzahlig, so muss mindestens eine der Nichtbasisvariablen N von Null verschieden sein, wegen der Ganzzahligkeit also mindestens 1, daraus folgt, dass die Ungleichung

$$\sum_{j \in N} x_j \geq 1$$

von allen ganzzahligen Punkten aber nicht von der gegenwärtigen Optimallösung erfüllt wird. Sie liefert also eine Schnittebene. Gomory und Hoffman haben 1963 ein Beispiel konstruiert, für das ein Verfahren mit diesen Schnittebenen nicht endlich terminiert.

Wir geben nun eine Klasse „guter Schnittebenen“ an.

(7.25) Lemma. *Gegeben sei ein ganzzahliges Programm* ($\mathrm{IP}^=$) *der Form* (7.3) *mit ganzzahligen Daten. Sei A_B eine Basis der LP-Relaxierung* ($\mathrm{LP}^=$)*, und sei $h \neq 0$ eine ganze Zahl, dann sind*

(a) $\sum_{j\in N}(\lfloor h\overline{a}_{ij}\rfloor - h\overline{a}_{ij})x_j \leq \lfloor h\overline{b}_i\rfloor - h\overline{b}_i \qquad \forall i \in B \qquad$ *und*

(b) $\sum_{j\in N}(\lfloor h\tilde{c}_j\rfloor - h\tilde{c}_j)x_j \leq \lfloor hc^*\rfloor - hc^*$

Schnittebenen, genannt fundamentale Schnittebenen*, falls $h\overline{b}_i$ bzw. hc^* nicht ganzzahlig sind. Dabei seien: c^* der Wert der Basislösung zu A_B von* ($\mathrm{LP}^=$)*, und (wie üblich) B die Indizes der Basisvariablen, N Indizes der Nichtbasisvariablen und*

$$\overline{A} = (\overline{a}_{ij}) = A_B^{-1}A_N, \quad \overline{b} = A_B^{-1}b, \quad \overline{c}^T = (c_N^T - c_B^T\overline{A}), \quad \tilde{c} = -\overline{c}. \qquad \triangle$$

Beweis. Jedes System $Ax = b$ hat bezüglich einer Basis A_B die Darstellung $x_B = A_B^{-1}b - A_B^{-1}A_Nx_N = \overline{b} - \overline{A}x_N$. (Die Basislösung bezüglich A_B ist gegeben durch $x_B = \overline{b}$, $x_N = 0$.) Es gilt also

$$x_i = \overline{b}_i - \overline{A}_{i.}x_N \quad \forall i \in B$$

bzw.

$$x_i + \sum_{j\in N}\overline{a}_{ij}x_j = \overline{b}_i \quad \forall i \in B. \tag{1}$$

Multiplizieren mit $h \in \mathbb{Z}$, $h \neq 0$, ergibt

$$hx_i + \sum_{j\in N}h\overline{a}_{ij}x_j = h\overline{b}_i. \tag{2}$$

Da bei Zulässigkeit $x_k \geq 0\ \forall k$ gelten muss, können wir die Koeffizienten auf der linken Seite von (2) abrunden und erhalten so aus der Gleichung (2) eine gültige Ungleichung

$$hx_i + \sum_{j\in N}\lfloor h\overline{a}_{ij}\rfloor x_j \leq h\overline{b}_i. \tag{3}$$

Für ganzzahlige Lösungen ergibt die linke Seite von (3) einen ganzzahligen Wert. Wenn wir somit die rechte Seite ebenfalls abrunden, bleibt die neu entstehende Ungleichung für alle $x \in \mathrm{IP}^=(A, b)$ gültig. Wir erhalten also die für $\mathrm{IP}^=(A, b)$ gültige Ungleichung

$$hx_i + \sum_{j\in N}\lfloor h\overline{a}_{ij}\rfloor x_j \leq \lfloor h\overline{b}_i\rfloor. \tag{4}$$

Ziehen wir nun (2) von (4) ab, so ergibt sich die gültige Ungleichung

$$\sum_{j\in N}(\lfloor h\overline{a}_{ij}\rfloor - h\overline{a}_{ij})x_j \leq \lfloor h\overline{b}_i\rfloor - h\overline{b}_i \quad \forall i \in B. \tag{5}$$

Ist nun die gegenwärtige Basislösung nicht ganzzahlig, so gibt es ein $i \in B$ mit $\overline{b}_i \notin \mathbb{Z}$. Ist $h\overline{b}_i \notin \mathbb{Z}$, so besagt die Ungleichung (5), dass die linke Seite kleiner oder gleich $\lfloor h\overline{b}_i \rfloor - h\overline{b}_i < 0$ sein muss. Da $x_N = 0$ für die Basislösung gilt, erfüllt die gegenwärtige Basislösung die Ungleichung (5) nicht, wird also abgeschnitten.

Der Zielfunktionswert $c_0 := c^T x$ bezüglich der Basis A_B lässt sich wie folgt berechnen:

$$c_0 = c^T x = c_B^T x_B + c_N^T x_N = c_B^T(\overline{b} - \overline{A}x_N) + c_N^T x_N = c_B^T \overline{b} + (c_N^T - c_B^T \overline{A})x_N = c_B^T \overline{b} + \overline{c}^T x_N,$$

wobei $\overline{c}$ die reduzierten Kosten sind. Setzen wir $\tilde{c} = -\overline{c}$, so ergibt dies

$$c_0 + \tilde{c}^T x_N = c_B^T \overline{b} =: c^*.$$

Multiplizieren mit $h \in \mathbb{Z}$ ergibt

$$hc_0 + \sum_{j \in N} h\tilde{c}_j x_j = hc^*. \tag{6}$$

Für zulässige Punkte x gilt $x_j \geq 0$, also liefert Abrunden

$$hc_0 + \sum_{j \in N} \lfloor h\tilde{c}_j \rfloor x_j \leq hc^*. \tag{7}$$

Ist x ganzzahlig, so ist c_0 ganzzahlig, weil $c \in \mathbb{Z}^n$ ist; also stehen auf der linken Seite von (7) nur ganze Zahlen, folglich kann die rechte Seite von (7) ebenfalls abgerundet werden, ohne die Gültigkeit für Punkte aus $\mathrm{IP}^=(A, b)$ zu verlieren. Es ergibt sich

$$hc_0 + \sum_{j \in N} \lfloor h\tilde{c}_j \rfloor x_j \leq \lfloor hc^* \rfloor. \tag{8}$$

Ziehen wir (6) von (8) ab, so erhalten wir

$$\sum_{j \in N} (\lfloor h\tilde{c}_j \rfloor - h\tilde{c}_j) x_j \leq \lfloor hc^* \rfloor - hc^*,$$

was zu zeigen war. □

Es ist klar, dass ein ganzzahliges Programm mit ganzzahligen Daten als Optimalwert eine ganze Zahl haben muss. Haben wir das zu ($\mathrm{IP}^=$) gehörige lineare Programm ($\mathrm{LP}^=$) gelöst, und hat dieses keinen ganzzahligen Optimalwert, so können wir mittels einer Schnittebene des Typs (b) die gegenwärtige Basislösung abschneiden. Ist der Optimalwert des LP ganzzahlig, aber die optimale Basislösung nicht ganzzahlig, so gibt es wegen $x_N = 0$ eine Basisvariable x_i mit $x_i = \overline{b}_i \notin \mathbb{Z}$. Dann können wir mit einer Ungleichung des Typs (a) wiederum die gegenwärtige Basislösung abschneiden.

Die obigen Überlegungen zeigen also, dass zu jeder nichtganzzahligen Basislösung eines LP eine Schnittebene existiert, die bezüglich aller ganzzahligen Punkte zulässig ist, die die betrachtete Basislösung abschneidet, und die – was wichtig ist – sehr einfach aus den LP-Daten abgeleitet werden kann. Im Prinzip kann dieses Verfahren unendlich lange dauern, wir werden aber zeigen, dass bereits für den Spezialfall $h = 1$ unter gewissen Voraussetzungen endliche Konvergenz gezeigt werden kann.

(7.26) Korollar. *Die Voraussetzungen seien wie in Lemma (7.25) Wir setzen*

$$\left.\begin{array}{ll} f_{ij} := \overline{a}_{ij} - \lfloor \overline{a}_{ij} \rfloor, & i \in B, j \in N \\ f_{0j} := -\overline{c}_j - \lfloor -\overline{c}_j \rfloor, & j \in N \\ f_{i0} := \overline{b}_i - \lfloor \overline{b}_i \rfloor & i \in B \\ f_{00} := c^* - \lfloor c^* \rfloor. & \end{array}\right\} \quad \textit{(diese rationalen Zahlen nennt man} \text{ gebrochene Teile}\textit{)}$$

Falls $f_{i0} \neq 0$ für $i = 0$ oder $i \in B$, dann ist der erste Gomory-Schnitt

$$\sum_{j \in N} -f_{ij} x_j \leq -f_{i0}$$

eine Ungleichung, die für alle Punkte aus $\mathrm{IP}^{=}(A, b)$ *zulässig ist und die die gegenwärtige Basislösung abschneidet.* △

Wir stellen nun einen Algorithmus zusammen, der auf der Basis des Simplex-Verfahrens ganzzahlige Programmierungsprobleme löst. In einer ersten Phase wird mit dem primalen Simplexalgorithmus eine optimale Lösung des zugehörigen LP produziert, dann werden Schnittebenen abgeleitet – und zwar erste Gomory-Schnitte – um die das Restriktionensystem erweitert wird. Mit dem dualen Simplexverfahren wird anschließend eine Reoptimierung vorgenommen. Der Algorithmus ist sehr ausführlich dargestellt.

(7.27) Algorithmus (Erster Gomory-Algorithmus).
Eingabe: $A \in \mathbb{Z}^{(m,n)}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$.
Ausgabe: Lösung des ganzzahligen Programms ($\mathrm{IP}^{=}$)

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

Zur einfacheren Darstellung des Algorithmus führen wir folgende Notation ein:

$a_{00} := c_0$	(gegenwärtiger Zielfunktionswert)
$a_{0j} := -c_j$, $j = 1, \ldots, n$	(negativer Wert der Zielfunktionskoeffizienten, um die Schnitte aus der Zielfunktion einfacher ableiten zu können)
$a_{i0} := b_i$, $i = 1, \ldots, m$	(rechte Seite)

Unser ($\mathrm{IP}^{=}$) lautet also in der neuen Notationsform

$$\begin{aligned} \max \quad a_{00} \ &= \ -\sum_{j=1}^{n} a_{0j} x_j \\ \sum_{j=1}^{n} a_{ij} x_j &= a_{i0}, \quad i = 1, \ldots, m \\ x_j &\geq 0 \text{ und ganzzahlig}, \quad j = 1, \ldots, n \end{aligned}$$

($\mathrm{LP}^{=}$) ist das obige Programm ohne Ganzzahligkeitsbedingungen.

1. In einer Phase 1 wird die Zulässigkeit des zugehörigen $(LP^{=})$ geprüft.
   1.1. Falls (LP$^{=}$) unzulässig ist, ist natürlich auch (IP$^{=}$) unzulässig. $\longrightarrow$ STOP.
   1.2. Ist (LP$^{=}$) zulässig, so endet die Phase 1 mit einer zulässigen Basis. Hierbei können u. U. einige linear abhängige Zeilen von $Ax = b$ gestrichen worden sein. Wir würden dann mit dem reduzierten System beginnen. Der Einfachheit halber nehmen wir an, dass A vollen Rang m hat. Wir erhalten also

$$B = (p_1, p_2, \ldots, p_m), \qquad N = (q_1, q_2, \ldots, q_{n-m})$$

   und somit eine Basismatrix A_B von A.

2. Wir berechnen

$$\begin{aligned}
&A_B^{-1} \\
&\overline{A} = A_B^{-1} A_N && (m, n-m)\text{-Matrix} \\
&\overline{a}_{.0} = A_B^{-1} b = A_B^{-1} a_{.0} && m\text{-Vektor} \\
&\overline{a}_{0.} = -(c_N^T - c_B^T A_B^{-1} A_N) && (n-m)\text{-Vektor} \\
&\qquad = a_{0,N}^T - a_{0,B}^T \overline{A} \\
&\overline{a}_{00} = c_B^T A_B^{-1} b = -a_{0,B}^T \overline{a}_{.0}.
\end{aligned}$$

Primaler Simplex-Algorithmus (Voraussetzung: zulässige Basis bekannt).

3. (Optimalitätsprüfung)
   Gilt $\overline{a}_{0j} \geq 0$ für $j = 1, \ldots, n-m$, so ist die gegenwärtige Basislösung optimal. Gehe zu Schritt 8. Andernfalls gehe zu Schritt 4.

4. (Bestimmung der Austauschspalte)
   Wähle einen Index $s \in \{1, \ldots, n-m\}$, so dass $\overline{a}_{0s} < 0$ gilt. (Hierzu haben wir mehrere Möglichkeiten kennengelernt; z. B. wähle s, so dass $\overline{a}_{0s}$ minimal ist.)

5. (Prüfung auf Beschränktheit des Optimums)
   Gilt $\overline{a}_{is} \leq 0$ für $i = 1, \ldots, m$, so ist das lineare Programm unbeschränkt, also ist entweder (IP$^{=}$) unbeschränkt oder hat keine zulässige Lösung. $\longrightarrow$ STOP.

6. (Bestimmung der Austauschzeile)
   6.1. Berechne $\lambda_0 := \min\left\{ \frac{\overline{a}_{i0}}{\overline{a}_{is}} \mid \overline{a}_{is} > 0, i = 1, \ldots, m \right\}$.
   6.2. Wähle einen Index $r \in \{1, \ldots, m\}$, so dass gilt

$$\frac{\overline{a}_{r0}}{\overline{a}_{rs}} = \lambda_0.$$

   (Hierzu haben wir ebenfalls mehrere Möglichkeiten kennengelernt, die die Endlichkeit des Verfahrens garantieren.)

7. (Basisaustausch, Pivotoperationen)

   7.1. Setze

$$B' := (p_1, p_2, \ldots, p_{r-1}, q_s, p_{r+1}, \ldots, p_m)$$
$$N' := (q_1, q_2, \ldots, q_{s-1}, p_r, q_{s+1}, \ldots, q_{n-m}).$$

   7.2. (Neuberechnung der erweiterten Matrix $\overline{A}$)
   Führe folgende Pivotoperationen durch:

   7.2.1. $\overline{\overline{a}}_{rs} := \dfrac{1}{\overline{a}_{rs}}$.

   7.2.2. $\overline{\overline{a}}_{rj} := \dfrac{\overline{a}_{rj}}{\overline{a}_{rs}}$, $j = 0, 1, \ldots, n-m$, $j \neq s$.

   7.2.3. $\overline{\overline{a}}_{is} := -\dfrac{\overline{a}_{is}}{\overline{a}_{rs}}$, $i = 0, 1, \ldots, m$, $i \neq r$.

   7.2.4. $\overline{\overline{a}}_{ij} := \overline{a}_{ij} - \dfrac{\overline{a}_{is}\overline{a}_{rj}}{\overline{a}_{rs}}$, $i = 0, 1, \ldots, m$, $i \neq r$, $j = 0, 1, \ldots, n-m$, $j \neq s$.

   7.3 Setze $B := B'$, $N := N'$ und $\overline{a}_{ij} := \overline{\overline{a}}_{ij}$, $i = 0, \ldots, m$, $j = 0, \ldots, n-m$ und gehe zu Schritt 3.

Ende des primalen Simplex-Algorithmus

Bestimmung der Schnittebenen, Optimalitätstest

8. (Prüfung auf Ganzzahligkeit der Lösung)
   Gilt $\overline{a}_{i0} \in \mathbb{Z}$ für $i = 0, 1, \ldots, m$, so ist die optimale Lösung des gegenwärtigen LP ganzzahlig und folglich auch eine optimale Lösung von (IP$^=$).
   Ausgabe: Zielfunktionswert $\overline{a}_{00}$, Optimallösung

$$x_i = 0 \quad \forall i \in N$$
$$x_i = \overline{a}_{i0} \quad \forall i \in B$$

   STOP.
   Andernfalls gehe zu Schritt 9.

9. (Ableitung einer neuen Restriktion, Schnittebenengenerierung)
   Wähle ein $i \in \{0, 1, \ldots, m\}$ so dass $\overline{a}_{i0} \notin \mathbb{Z}$ (hierfür kann man sich mehrere Strategien überlegen), und setze für $j = 0, 1, \ldots, n-m$

$$\overline{a}_{m+1,j} := -(\overline{a}_{ij} - \lfloor \overline{a}_{ij} \rfloor).$$

   Füge die Zeile

$$\overline{a}_{m+1,0},\ \overline{a}_{m+1,1},\ \ldots,\ \overline{a}_{m+1,n-m}$$

   als $(m+1)$-te Zeile zum gegenwärtigen Tableau hinzu.

10. (Basiserweiterung)
Nimm die zur neu hinzugefügten Ungleichung gehörige Schlupfvariable x_{n+1} mit dem Wert $\overline{a}_{m+1,0}$ in die Basis auf, d. h. setze $B = (p_1, p_2, \ldots, p_m, n+1)$ und setze

$$m := m + 1, \quad n := n + 1.$$

Gehe zu Schritt 11.

Dualer Simplex-Algorithmus (Voraussetzung: dual zulässige Basis bekannt)

Wir haben das duale Simplex-Verfahren und das Vorgehen bei Hinzufügung einer neuen Restriktion, falls eine Basislösung des linearen Programms bekannt ist, bereits behandelt. Das zum (primalen) linearen Programm ($\mathrm{LP}^{=}$)

$$\max c^T x,\ Ax = b, x \geq 0$$

duale Programm lautet $\min u^T b,\ u^T A \geq c^T$. Die zu einer Basis A_B von A gehörige Basislösung ist $x_B = A_B^{-1} b$, $x_N = 0$. Falls $x_B \geq 0$ ist, heißt diese Basislösung primal zulässig, da sie eine zulässige Lösung von ($\mathrm{LP}^{=}$) darstellt. Eine Basislösung heißt dual zulässig, wenn der Vektor $c_B^T A_B^{-1}$ eine zulässige Lösung des dualen Programms ist. Eine Basislösung ist also genau dann dual zulässig, wenn $c_B^T A_B^{-1} A \geq c^T$ gilt, bzw. (1) $c_B^T A_B^{-1} A_B \geq c_B^T$ und (2) $c_B^T A_B^{-1} A_N \geq c_N^T$ gilt. Bedingung (1) ist natürlich immer erfüllt, Bedingung (2) ist erfüllt, wenn $\overline{c}^T := c_N^T - c_B^T A_B^{-1} A_N \leq 0$ ist. Dies ist aber gerade die primale Optimalitätsbedingung (die reduzierten Kosten sind nicht positiv). Es folgt, dass eine Basislösung primal und dual zulässig ist genau dann, wenn sie optimal ist.

Erweitern wir unser Restriktionssystem um eine Schnittebene, so können wir durch Aufnahme der Schlupfvariablen in die Basis sofort eine neue Basis bzw. Basislösung angeben. Da der Wert der neuen Basisvariablen nach Konstruktion negativ ist, ist die Basislösung nicht primal zulässig. Sie ist jedoch dual zulässig, da sich die reduzierten Kosten nicht geändert haben. Wir haben somit eine Startbasis zur Ausführung des dualen Simplexverfahrens.

11. (Optimalitätstest, d. h. Test auf primale Zulässigkeit)
Gilt $\overline{a}_{i0} \geq 0$, $i = 1, \ldots, m$, so ist eine optimale Lösung des gegenwärtigen linearen Programms gefunden; gehe zu Schritt 8.
Andernfalls gehe zu Schritt 12.

12. (Bestimmung der in die duale Basis eintretenden Variablen)
Wähle einen Index $r \in \{1, \ldots, m\}$ mit $\overline{a}_{r0} < 0$. (Hierzu gibt es wiederum mehrere Strategien, z. B. $\overline{a}_{r0}$ minimal.)

13. (Prüfung auf Beschränktheit des Optimums)
Gilt $\overline{a}_{rj} \geq 0$ für $j = 1, \ldots, n - m$, so hat das duale Programm keine endliche Optimallösung, d. h. das primale Programm hat überhaupt keine Lösung, also gilt $\mathrm{IP}^{=}(A, b) = \emptyset$. $\longrightarrow$ STOP.

14. (Bestimmung der aus der dualen Basis austretenden Variablen)

    14.1. Berechne $\lambda_0 := \max\left\{ \frac{\overline{a}_{0j}}{\overline{a}_{rj}} \mid \overline{a}_{rj} < 0, j = 1, \ldots, n-m \right\}$.

    14.2. Wähle einen Index $s \in \{1, \ldots, n-m\}$, so dass gilt

    $$\frac{\overline{a}_{0s}}{\overline{a}_{rs}} = \lambda_0$$

    (mehrere Strategien möglich, falls s nicht eindeutig bestimmt ist).

15. (Basisaustausch, Pivotoperation)

    15.1. Setze

    $$B' := (p_1, p_2, \ldots, p_{r-1}, q_s, p_{r+1}, \ldots, p_m)$$
    $$N' := (q_1, q_2, \ldots, q_{s-1}, p_r, q_{s+1}, \ldots, q_{n-m}).$$

    15.2. (Neuberechnung des Tableaus)

    15.2.1. $\overline{\overline{a}}_{rs} := \dfrac{1}{\overline{a}_{rs}}$.

    15.2.2. $\overline{\overline{a}}_{rj} := \dfrac{\overline{a}_{rj}}{\overline{a}_{rs}}$, $j = 0, 1, \ldots, n-m$, $j \neq s$.

    15.2.3. $\overline{\overline{a}}_{is} := -\dfrac{\overline{a}_{is}}{\overline{a}_{rs}}$, $i = 0, 1, \ldots, m$, $i \neq r$.

    15.2.4. $\overline{\overline{a}}_{ij} := \overline{a}_{ij} - \dfrac{\overline{a}_{is}\overline{a}_{rj}}{\overline{a}_{rs}}$, $i = 0, 1, \ldots, m$, $i \neq r$, $j = 0, 1, \ldots, n-m$, $j \neq s$.

    15.3. Setze $B := B'$, $N := N'$ und $\overline{a}_{ij} := \overline{\overline{a}}_{ij}$ für $i = 0, \ldots, m$, $j = 0, \ldots, n-m$ und gehe zu Schritt 11.

Ende des dualen Simplex-Algorithmus. △

(7.28) Beispiel. Wir lösen das folgende ganzzahlige Programm mit dem ersten Gomory-Algorithmus. Die Lösungsmenge ist in Abbildung 7.7 dargestellt.

$$\begin{array}{cc} & \max\ x_1 + 2x_2 \\ x_3 \longrightarrow & x_1 \leq 4 \\ x_4 \longrightarrow & 2x_1 + x_2 \leq 10 \\ x_5 \longrightarrow & -x_1 + x_2 \leq 5 \\ & x_1, x_2 \geq 0 \\ \uparrow & x_1, x_2 \text{ ganzzahlig} \\ \text{Schlupfvariable} & \end{array}$$

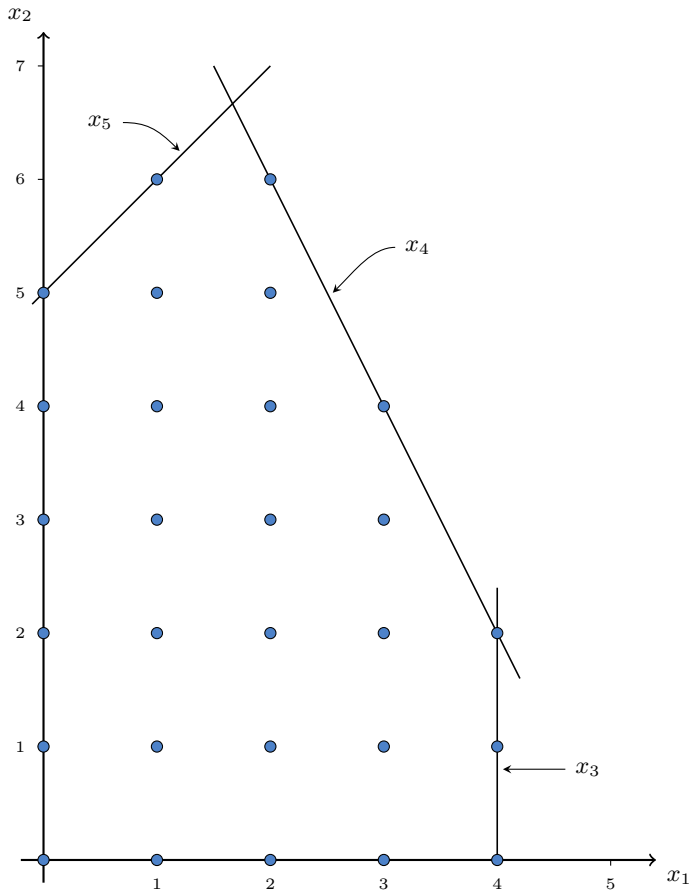


Abbildung 7.7: Lösungsmenge des IP in Beispiel (7.28) mit zu den entsprechenden Ungleichungen korrespondierenden Schlupfvariablen

(a) Anfangsbasis aus Schlupfvariablen, $B = (3, 4, 5)$, $N = (1, 2)$.
1. Tableau:

		1	*2*	$\leftarrow N$
	0	-1	-2	
3	4	1	0	
4	10	2	1	
5	5	-1	$\boxed{1}$	
$\uparrow$				
B				

$s = 2$
$r = 3$

Eine Pivotoperation ergibt das 2. Tableau:

		1	*5*	
	10	−3	2	$s = 1$
3	4	1	0	$r = 2$
4	5	$\boxed{3}$	−1	
2	5	−1	1	

3. Tableau:

		4	*5*	
	15	1	1	Optimalitätstest erfüllt
3	$7/3$	$-1/3$	$1/3$	
1	$5/3$	$1/3$	$-1/3$	
2	$20/3$	$1/3$	$2/3$	

Optimallösung:

$$x_1 = \frac{5}{3}, \qquad x_2 = \frac{20}{3}, \qquad c^* = 15.$$

Ende des primalen Simplexverfahrens.

(b) **Schnittebenenbestimmung**
Außer der Zielfunktionszeile können alle anderen Zeilen zur Schnittebenenberechnung benutzt werden, da $\overline{a}_{i0} \notin \mathbb{Z}$, $i = 1, 2, 3$. Wir bestimmen alle drei Schnittebenen (Kandidaten für die neue vierte Zeile des Tableaus).

1. Zeile: $\overline{a}_{40} = -(\frac{7}{3} - \frac{6}{3}) = -\frac{1}{3}$, $\overline{a}_{41} = -(\frac{1}{3} - \frac{-3}{3}) = -\frac{2}{3}$, $\overline{a}_{42} = -(\frac{1}{3} - 0) = -\frac{1}{3}$.
Daraus resultiert die Schnittebene

$$-\frac{2}{3}x_4 - \frac{1}{3}x_5 \leq -\frac{1}{3}.$$

In den ursprünglichen Variablen – wenn wir $x_4 = 10 - 2x_1 - x_2$ und $x_5 = 5 + x_1 - x_2$ substituieren – ergibt dies:

$$\begin{aligned} -\frac{2}{3}(10 - 2x_1 - x_2) - \frac{1}{3}(5 + x_1 - x_2) \leq -\frac{1}{3} \quad &\Longleftrightarrow \\ -20 + 4x_1 + 2x_2 - 5 - x_1 + x_2 \leq -1 \quad &\Longleftrightarrow \\ 3x_1 + 3x_2 \leq 24 \end{aligned}$$

Unsere Schnittebene ist also äquivalent zur Ungleichung

$$x_1 + x_2 \leq 8.$$

2. Zeile: $\overline{a}_{40} = -\frac{2}{3}$, $\overline{a}_{41} = -\frac{1}{3}$, $\overline{a}_{42} = -\frac{2}{3}$.

$$-\frac{1}{3}x_4 - \frac{2}{3}x_5 \leq -\frac{2}{3}$$

bzw. in ursprünglichen Variablen

$$\begin{aligned} -\frac{1}{3}(10 - 2x_1 - x_2) - \frac{2}{3}(5 + x_1 - x_2) \leq -\frac{2}{3} \quad &\Longleftrightarrow \\ -10 + 2x_1 + x_2 - 10 - 2x_1 + 2x_2 \leq -2 \quad &\Longleftrightarrow \\ 3x_2 \leq 18, \end{aligned}$$

das heißt, die Schnittebene ist äquivalent zu:

$$x_2 \leq 6.$$

3. Zeile: $\overline{a}_{40} = -\frac{2}{3}$, $\overline{a}_{41} = -\frac{1}{3}$, $\overline{a}_{42} = -\frac{2}{3}$, also

$$-\frac{1}{3}x_4 - \frac{2}{3}x_5 \leq -\frac{2}{3}.$$

Damit ergibt sich dieselbe Schnittebene wie bei der Ableitung aus der zweiten Zeile.

Die beiden auf diese Weise gefundenen Schnittebenen sind in Abbildung 7.8 dargestellt. An dem obigen Beispiel kann man sehen, dass die Gomory-Schnitte verschieden „tief“ sein können. Der Schnitt aus der zweiten Zeile ist tiefer, weil der Durchschnitt von P mit dem zugehörigen Halbraum echt im Durchschnitt von P mit dem aus der ersten Zeile abgeleiteten Halbraum enthalten ist. Die Schnittebene aus der zweiten Zeile ergibt zusammen mit den ursprünglichen Ungleichungen sogar die konvexe Hülle der ganzzahligen Punkte. Daraus folgt, dass bei Hinzufügung dieses Schnittes das ganzzahlige Optimum erreicht wird.

Bei normalem Durchlauf des Algorithmus hätten wir wahrscheinlich den ersten Schnitt gewählt, deshalb fügen wir diesen zusammen mit der neuen Schlupfvariablen x_6 zu unserem Tableau hinzu.

Neues Tableau:

		4	*5*	
	15	1	1	
3	$^7/_3$	$-^1/_3$	$^1/_3$	
1	$^5/_3$	$^1/_3$	$-^1/_3$	
2	$^{20}/_3$	$^1/_3$	$^2/_3$	
6	$-^1/_3$	$\boxed{-^2/_3}$	$-^1/_3$	← neue Zeile, Schlupfvariable x_6 in der Basis

(c) Duales Simplexverfahren angewendet auf das obige Tableau

$$r = 4, \qquad \lambda_0 = -\frac{3}{2}, \qquad s = 1.$$

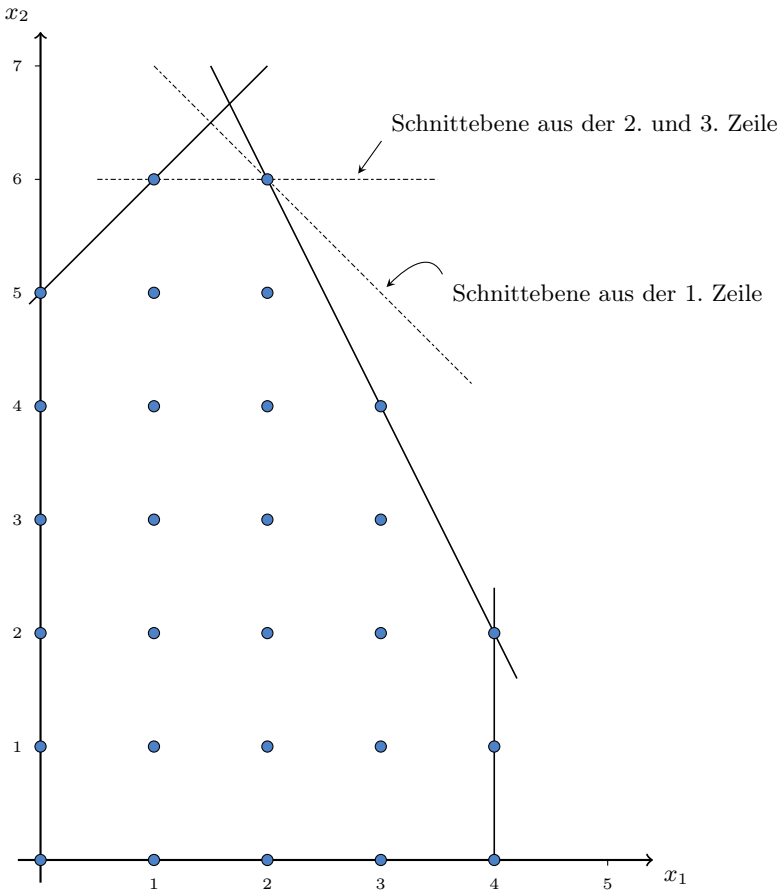


Abbildung 7.8: Zwei Schnittebenen aus der ersten Runde

Die Pivotoperation ergibt:

		6	*5*
	$29/2$	$3/2$	$1/2$
3	$5/2$	$-1/2$	$1/2$
1	$3/2$	$1/2$	$-1/2$
2	$13/2$	$1/2$	$1/2$
4	$1/2$	$-3/2$	$1/2$

Dieses Tableau ist optimal, jedoch nicht ganzzahlig.

(d) Ableitung eines neuen Schnittes aus der Zielfunktion: $\overline{a}_{50} = -\frac{1}{2}, \overline{a}_{51} = -\frac{1}{2}, \overline{a}_{52} = -\frac{1}{2}$. Daraus ergibt sich

$$-\frac{1}{2}x_5 - \frac{1}{2}x_6 \leq -\frac{1}{2}.$$

Wegen $x_5 = 5 + x_1 - x_2$ und $x_6 = 8 - x_1 - x_2$ sieht diese Ungleichung in den

ursprünglichen Variablen wie folgt aus

$$x_2 \leq 6.$$

Diese Ungleichung hätten wir bereits nach Lösung des Anfangs-LP mit dem primalen Verfahren aus der 2. oder 3. Zeile erhalten können.

Neues Tableau:

		6	*5*	
	$29/2$	$3/2$	$1/2$	
3	$5/2$	$-1/2$	$1/2$	
1	$3/2$	$1/2$	$-1/2$	
2	$13/2$	$1/2$	$1/2$	
4	$1/2$	$-3/2$	$1/2$	
7	$-1/2$	$-1/2$	$\boxed{-1/2}$	$\leftarrow$ neue Zeile, Schlupfvariable x_7

(e) Erneute Anwendung des dualen Simplexverfahrens:

		6	*7*	
	14	1	1	
3	2	-1	1	
1	2	1	-1	$r = 5$
2	6	0	1	$s = 2$
4	0	-2	1	
5	1	1	-2	
	$\uparrow$			

entartete Basislösung, wegen $x_4 = 0$

Optimallösung gefunden:

$$x_1 = 2, \qquad x_2 = 6, \qquad c^* = 14 \qquad \triangle$$

Der Gesamtablauf des Verfahrens ist in Abbildung 7.9 zusammengefasst.

(7.29) Bemerkung (Beobachtungen zum ersten Gomory-Algorithmus).

(a) Zeigt das primale Simplexverfahren in Schritt 5 die Unbeschränktheit der LP-Relaxierung an, so ist $(\mathrm{IP}^=)$ nicht sinnvoll lösbar, d. h. entweder unbeschränkt oder $\mathrm{IP}^=(A, b) = \emptyset$.

(b) Wird in Schritt 13 die Unbeschränktheit des dualen Programms zum gegenwärtigen LP (dieses ist das um Schnittebenen erweiterte ursprüngliche LP) festgestellt, so ist $\mathrm{IP}^=(A, b) = \emptyset$.

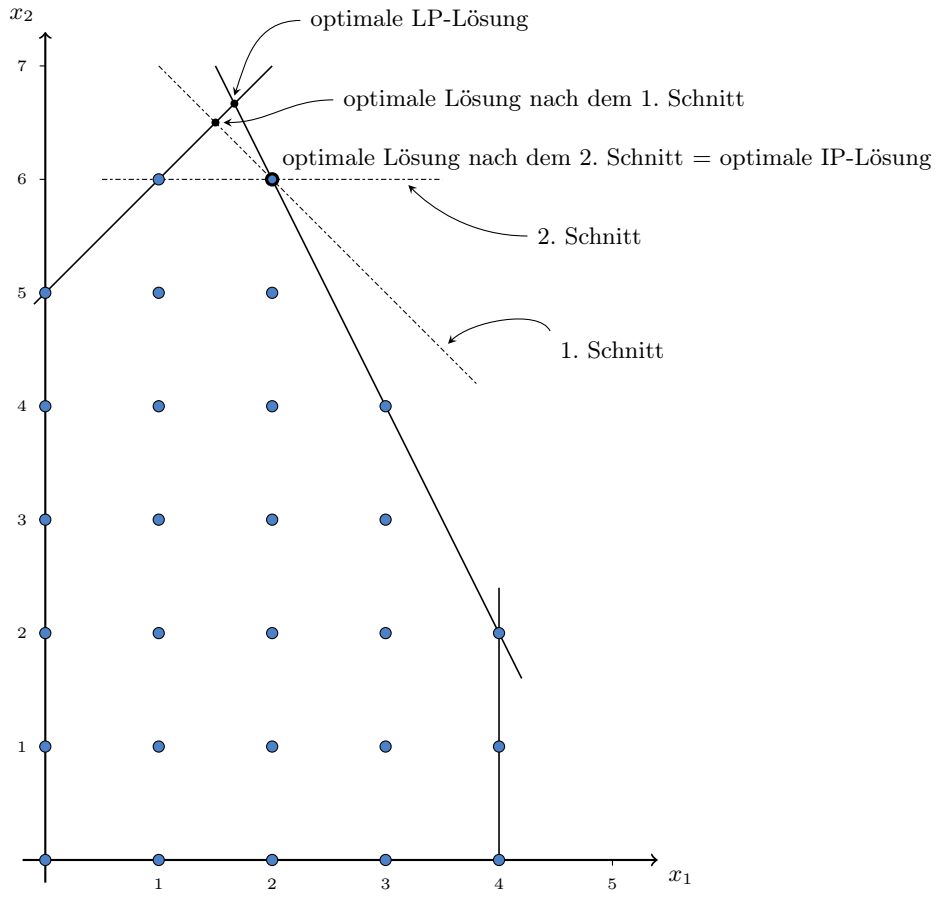


Abbildung 7.9: Hinzugefügte Schnitte während des Gomory-Verfahrens und optimale Lösungen in den Zwischenschritten

(c) Obwohl an zwei Stellen des Algorithmus die Unzulässigkeit von ($IP^{=}$) festgestellt werden kann, gibt es keine Garantie dafür, dass der Algorithmus in dieser Form die Unzulässigkeit auch beweist. Wir können diesen Mangel mit der Einführung von Schranken, siehe Folgerung (7.15) (oder (7.17)), reparieren und einen Zusatzschritt formulieren.

(d) Falls das primale oder duale Simplex-Verfahren eine nicht ganzzahlige Lösung liefert, so kann das Schnittebenen-Unterprogramm theoretisch immer einen Gomory-Schnitt finden, der diese Lösung abschneidet. Wir schreiben hier „theoretisch“, da bei praktischen Rechnungen mit reeller floating-point-Arithmetik aufgrund von Rundefehlern und Toleranzgrößen die Entscheidung, ob eine Größe ganzzahlig ist oder nicht, schwierig zu fällen ist. △

Bevor wir uns dem Beweis der Korrektheit von Algorithmus (7.27) zuwenden, formulieren wir einige Zusatzregeln – die allerdings nur von theoretischem Interesse sind.

(7.30) Bemerkung (Sonderregeln zum ersten Gomory-Algorithmus).

(a) Ein Vektor a heißt *lexikographisch größer* als ein Vektor b, wenn entweder $a_1 > b_1$ gilt oder ein k existiert mit $1 < k \leq n$, $a_i = b_i$ für $i = 1, \ldots, k-1$ und $a_k > b_k$. Die Spalten des Simplextableaus (inklusive der 0-ten Zeile) bezeichnen wir wie immer mit $\overline{A}_{\cdot j}$, d. h.

$$\overline{A}_{\cdot j} = \begin{pmatrix} \overline{a}_{0j} \\ \overline{a}_{1j} \\ \vdots \\ \overline{a}_{mj} \end{pmatrix}, \quad j = 1, \ldots, n-m.$$

Wir gestalten unser Verfahren so, dass alle Spalten $\overline{A}_{\cdot j}$, $j = 1, \ldots, n-m$ nach Ende des primalen Simplexverfahrens lexikographisch positiv sind und weiterhin bleiben.

Ist die optimale LP-Lösung nicht *dual entartet*, d. h. sind alle (negativen) reduzierten Kosten $\overline{a}_{0j}$ positiv, so ist die lexikographische Positivität offenbar gegeben. Da aufgrund der Optimalität $\overline{a}_{0j} \geq 0$, $j = 1, \ldots, n-m$ gilt, kann kein $\overline{a}_{0j} < 0$ sein. Probleme treten auf, wenn einige $\overline{a}_{0j} = 0$ sind. Durch Umstellen der Zeilen des Tableaus kann u. U. lexikographische Positivität erreicht werden. Falls dies nicht geht, dann fügen wir eine Ungleichung hinzu, die die lexikographische Positivität gewährleistet und keine Ecke von $\mathrm{conv}(\mathrm{IP}^{=}(A,b))$ abschneidet. Diese Ungleichung kann ganzzahlige Punkte abschneiden, lässt aber mindestens eine der ganzzahligen Optimallösungen, falls überhaupt welche existieren, übrig. Eine Ungleichung dieses Typs ist z. B.

$$\sum_{j \in N} x_j \leq (n-m)(n+1)2^{4n^3\langle A,b\rangle}$$

was aus (7.15) folgt. Diese Ungleichung fügen wir dann als erste Zeile unterhalb der Zielfunktion in unser Tableau ein, wobei eine neue Schlupfvariable addiert werden muss. Die Schlupfvariable geht mit dem Wert der rechten Seite in die Basis. Nunmehr ist unser Tableau lexikographisch positiv, da $\overline{a}_{0j} \geq 0$ und $\overline{a}_{1j} = 1$ für $j = 1, \ldots, n-m$.

(b) Zur Beibehaltung der lexikographischen Positivität muss eine besondere Spaltenauswahlregel benutzt werden.

> 14.2. Sei $S = \{j \in \{1, \ldots, n-m\} \mid \frac{\overline{a}_{0j}}{\overline{a}_{rj}} = \lambda_0, \overline{a}_{rj} < 0\}$.
> Wähle denjenigen Index $s \in S$, so dass $\frac{1}{\overline{a}_{rs}}\overline{A}_{\cdot s}$ der lexikographisch größte der Vektoren $\frac{1}{\overline{a}_{rj}}\overline{A}_{\cdot j}$, $j \in S$ ist.

Hausaufgabe: Ist r eine in Schritt 12 gewählte Pivotzeile und s eine mit obiger Regel bestimmte Pivotspalte, dann ist das Tableau nach der Pivotoperation weiterhin lexikographisch positiv.

(c) Wir müssen noch gewährleisten, dass der Algorithmus erkennt, dass $(\mathrm{IP}^{=})$ unbeschränkt oder $\mathrm{IP}^{=}(A,b)$ leer ist. Hierzu gibt es zwei Möglichkeiten. Aus den Folgerungen (7.15), (7.17) kann man ableiten, dass der Absolutwert des Optimalwertes

nicht größer als $2^{\langle c\rangle+5n^4\langle A,b\rangle-2n}$ ist bzw. dass die Menge der Optimallösungen von $(IP^{=})$ im Würfel

$$W := \{x \in \mathbb{Z}^n \mid -(n+1)\cdot 2^{4n^3\langle A,b\rangle} \leq x_i \leq (n+1)\cdot 2^{4n^3\langle A,b\rangle}\}$$

liegt. Wir können daher entweder die Restriktion von W zu unserem ursprünglich ganzzahligen Programm hinzufügen (auf diese Weise haben wir allgemeine ganzzahlige Programme auf beschränkte ganzzahlige Programme reduziert) oder wir bauen in Schritt 8 die folgende Abfrage ein:

> **Zusatz zu Schritt 8:**
> Gilt für den gegenwärtigen Zielfunktionswert $\overline{a}_{00} < -2^{\langle c\rangle+5n^4\langle A,b\rangle-2n}$, so besitzt das ganzzahlige Programm keine zulässige Lösung. $\longrightarrow$ STOP.

Es muss darauf hingewiesen werden, dass dieses Abbruchkriterium, die Entdeckung der Unzulässigkeit des ($IP^{=}$) nur theoretisch gewährleistet, denn die obige Schranke ist i. A. ungeheuer groß und kaum auf dem Rechner darstellbar. Der Gomory-Algorithmus dürfte selbst bei sehr kleinen Beispielen Tausende von Jahren brauchen, um mit diesem Kriterium abzubrechen.

(d) Ferner müssen wir noch gewährleisten, dass die Schnittebenen in einer bestimmten Weise ausgewählt werden, und zwar leiten wir den ersten Gomory-Schnitt immer aus der am weitesten oben stehenden Zeile des Tableaus ab:

> **Zusatz zu Schritt 9:**
> Wähle den kleinsten Zeilenindex $i \in \{0, 1, \ldots, m\}$ mit $\overline{a}_{i0} \notin \mathbb{Z}$.

(e) Wir wollen nun noch eine Regel einführen, die garantiert, dass die Zahl unserer Zeilen, d. h. die Zahl der Restriktionen nicht beliebig groß werden kann. Angenommen nach Ende des dualen Simplexverfahrens ist eine Schlupfvariable x_{pi}, die zu einer Gomory-Schnittebene gehört, Basisvariable. Dann hat x_{pi} den Wert $\overline{a}_{i0} \geq 0$. Ist $x_{pi} > 0$, so bedeutet das, dass die Schnittebene bezüglich der gegenwärtigen Lösung nicht bindend ist. Streichen wir die Schnittebene aus unserem gegenwärtigen Restriktionensystem und eliminieren wir die Schlupfvariable, so ist unsere gegenwärtige Lösung eine optimale Basislösung des neuen Systems. Ist $x_{pi} = 0$, so ist die Basislösung entartet; zwar liegt die gegenwärtige Lösung auf der Schnittebene, jedoch können wir die Schnittebene weglassen, ohne die Optimalität zu verlieren. Das Streichen einer Zeile des Tableaus kann jedoch dazu führen, dass das Tableau nicht mehr lexikographisch positiv ist.
Man kann aber zeigen (siehe (Garfinkel und Nemhauser, 1972, S. 163–164)): Wird bei einer Pivotoperation eine Variable, die Schlupfvariable einer Schnittebene ist, in die Basis aufgenommen, so können die Pivotzeilen nach der Pivotoperation gestrichen und die Schlupfvariable eliminiert werden, so dass das reduzierte Tableau weiterhin lexikographisch positiv ist.

Zusatz zu Schritt 15:

15.2.5. Ist die neue Basisvariable die Schlupfvariable einer Schnittebene, so streichen wir die r-te Zeile des Tableaus und das r-te Element von B'. Anschließend setzen wir $m = m-1$, $n = n-1$ und numerieren die Zeilen mit Index $\geq r+1$ und die Schlupfvariablen neu.

Diese Zusatzregel garantiert uns, dass nur ursprüngliche Strukturvariable in der Basis sind, also kann unser Tableau niemals mehr als n Zeilen (plus Zielfunktion) enthalten. △

Wir werden anschließend zeigen, dass mit diesen Zusatzregeln Konvergenz des Gomory-Verfahrens bewiesen werden kann. Bei echten Implementationen wird jedoch kaum die lexikographische Spaltenauswahlregel (b) benutzt, da andere Regeln viel schneller zu einer Optimallösung des dualen Simplexverfahrens führen. Konvergiert das Gomory-Verfahren mit den anderen Regeln nicht, so konvergiert es mit der lexikographischen Regel in der Praxis auch nicht. Bei den Schnittebenen nimmt man meistens solche, die einen besonders „tiefen“ Schnitt ermöglichen und nicht die erste mögliche. In der Praxis ist das Weglassen von Restriktionen ebenfalls umstritten, wobei hier keine „guten“ Strategien angegeben werden können.

(7.31) Satz. *Sei* $(\mathrm{IP}^{=})$ *ein ganzzahliges Programm der Form* (7.3) *mit ganzzahligen Daten, dann liefert der erste Gomory-Algorithmus (7.27) mit den Zusatzregeln (7.30) nach endlich vielen Schritten eine ganzzahlige Lösung, oder er weist nach endlich vielen Schritten nach, dass entweder* $(\mathrm{IP}^{=})$ *unbeschränkt oder* $\mathrm{IP}^{=}(A,b)$ *leer ist.* △

Beweis. Der Beweis verläuft wie folgt: Wir wissen, dass bei jedem Schritt des dualen Simplexalgorithmus der Zielfunktionswert abnimmt oder gleich bleibt. Also kann nach einem Durchlauf des dualen Simplexalgorithmus der Zielfunktionswert nicht größer geworden sein. Wir zeigen nun zuerst, dass bei Ausführung des Gomory-Verfahrens der Zielfunktionswert nach endlich vielen Schritten einen festen ganzzahligen Wert annimmt und sich der Zielfunktionwert von da an nicht mehr ändert. Dann zeigen wir, dass nach weiteren endlich vielen Schritten das Tableauelement $\overline{a}_{10}$ einen festen ganzzahligen Wert annimmt und diesen Wert fortan beibehält. Wir führen diese Argumentation weiter und zeigen, dass das gleiche für $\overline{a}_{20}, \overline{a}_{30}, \ldots$ gilt. Aufgrund der „Streichregel“ (Zusatz zu Schritt 15) wissen wir, dass unser Tableau niemals mehr als n Zeilen enthalten kann. Da nach endlich vielen Schritten $\overline{a}_{10}$ ganzzahlig ist und der Wert sich nicht mehr verändert, da nach weiteren endlich vielen Schritten das gleiche für $\overline{a}_{20}$ gilt usw., ist nach endlich vielen Schritten das letzte Tableauelement $\overline{a}_{m0}$ $(m \leq n)$ ganzzahlig. Damit ist dann eine ganzzahlige Lösung des $(\mathrm{IP}^{=})$ gefunden.

Liefert der Gomory-Algorithmus nach endlich vielen Durchläufen die Unbeschränktheit oder die Unzulässigkeit von $(\mathrm{IP}^{=})$, so ist der zweite Teil unserer Behauptung gezeigt.

Damit der Gomory-Algorithmus nicht nach endlich vielen Schritten abbricht, müssen also das primale LP und anschließend jedes duale LP ein endliches Optimum haben. Aufgrund der Zusatzregel zu Schritt 8 kann der Zielfunktionswert zum dualen Programm

niemals kleiner werden als $-2^{\langle c\rangle+5n^4\langle A,b\rangle-2n}$, daraus folgt, dass die Folge $(\overline{a}_{00}^k)_{k\in\mathbb{N}}$ der Zielfunktionswerte nach dem k-ten Durchlauf des dualen Programms eine nach unten beschränkte, monoton fallende Folge ist. Diese Folge konvergiert also.

Wir untersuchen nun, wie sich der Algorithmus verhält, wenn der Optimalwert $\overline{a}_{00}^k$ des k-ten dualen Programms nicht ganzzahlig ist.

Die Zusatzregel (d) zu Schritt 9 besagt, dass die neue Schnittebene aus der Zielfunktionszeile abgeleitet werden muss. Fügen wir die Schnittebene zum Tableau hinzu, so wird die neue Schlupfvariable mit dem Wert $\overline{a}_{m+1,0} = \lfloor a_{00}^k\rfloor - a_{00}^k < 0$ Basisvariable. Da das Tableau nach Ende des dualen Verfahrens primal und dual zulässig ist, d. h. $\overline{a}_{i0} \geq 0$, $i = 1,\ldots,m$, und $\overline{a}_{0j} \geq 0$, $j = 1,\ldots,n$, ist die neue Basisvariable die einzige mit negativem Wert. Folglich wird die neu hinzugefügte Zeile $m+1$ in Schritt 12 als Pivotzeile gewählt. Sei s die in Schritt 14 gewählte Pivotspalte, dann gilt $\overline{a}_{m+1,s} < 0$. Weiterhin impliziert $\overline{a}_{0s} \geq 0$, dass $\overline{a}_{0s} \geq \overline{a}_{0s} - \lfloor\overline{a}_{0s}\rfloor = f_{0s} = -\overline{a}_{m+1,s}$ gilt (siehe Schritt 9). Daraus folgt $\frac{\overline{a}_{0s}}{-\overline{a}_{m+1,s}} \geq 1$. Die Pivotoperation in Schritt 15 liefert nunmehr:

$$\begin{aligned}\overline{\overline{a}}_{00} &= \overline{a}_{00}^k - \frac{\overline{a}_{0s}\overline{a}_{m+1,0}}{\overline{a}_{m+1,s}} = \overline{a}_{00}^k + \frac{\overline{a}_{0s}}{-\overline{a}_{m+1,s}}\overline{a}_{m+1,0}\\ &\leq \overline{a}_{00}^k + \overline{a}_{m+1,0} = \overline{a}_{00}^k + \lfloor\overline{a}_{00}^k\rfloor - \overline{a}_{00}^k\\ &= \lfloor\overline{a}_{00}^k\rfloor.\end{aligned}$$

Also ergibt die erste Pivotoperation nach Hinzufügung der aus der Zielfunktion abgeleiteten Schnittebene: $\overline{\overline{a}}_{00} \leq \lfloor\overline{a}_{00}^k\rfloor$. Da im weiteren Verlauf bis zur nächsten Optimallösung $\overline{a}_{00}^{k+1}$ der Zielfunktionswert nicht größer werden kann, gilt nach Ende des $(k+1)$-ten Durchlaufs des dualen Programms

$$\overline{a}_{00}^{k+1} \leq \lfloor\overline{a}_{00}^k\rfloor.$$

Nähme der Zielfunktionswert $\overline{a}_{00}^k$ nach Beendigung des dualen Verfahrens unendlich oft einen nicht ganzzahligen Wert an, so wäre nach der geraden abgeleiteten Beziehung die Folge $(\overline{a}_{00}^k)$ nach unten unbeschränkt. Aufgrund des Zusatzes zu Schritt 8 würde dann aber nach endlich vielen Schritten ein Abbruch erfolgen. Also ist die Folge $(\overline{a}_{00}^k)$ nur endlich oft nicht ganzzahlig. Da sie nach unten beschränkt ist, nimmt sie nach endlich vielen Schritten einen ganzzahligen Wert an und behält diesen Wert von da an bei, d. h. die Folge $(\overline{a}_{00}^k)$ ist nach endlich vielen Gliedern konstant und zwar mit einem ganzzahligen Wert.

Wir wollen uns nun überlegen, dass $\overline{a}_{10}$ nach endlich vielen Schritten einen festen ganzzahligen Wert annimmt und diesen Wert von da an beibehält. Aufgrund der obigen Überlegungen können wir voraussetzen, dass $\overline{a}_{00}$ nach dem k_0-ten Durchlauf des dualen Simplexalgorithmus seinen endgültigen ganzzahligen Wert erreicht hat. In der Optimallösung des k_0-ten dualen Programms hat das Tableauelement $\overline{a}_{10}$ einen nicht-negativen Wert (Schritt 11).

Wir zeigen nun, dass nach Beendigung des k_0-ten Durchlaufs bei jeder weiteren Pivotoperation der Wert von $\overline{a}_{10}$ nicht-negativ bleibt, nicht größer wird und dass – falls die Schnittebene aus der ersten Zeile abgeleitet wird – nach der ersten anschließenden Pivotoperation $\overline{\overline{a}}_{10} \leq \lfloor\overline{a}_{10}\rfloor$ gilt.

Nehmen wir an, dass in den Schritten 12 und 14 ein Pivotelement $\overline{a}_{rs} < 0$ ausgewählt wurde. Angenommen $\overline{a}_{0s} > 0$, dann folgt aus der Pivotformel 15.2.4. (wegen $\overline{a}_{r0} < 0$, $\overline{a}_{rs} < 0$, dass der Zielfunktionswert $\overline{a}_{00}$ abnimmt. Nach Voraussetzung ist aber $\overline{a}_{00}$ konstant, also gilt $\overline{a}_{0s} = 0$. Folglich muss aufgrund der lexikographischen Positivität des Tableaus $\overline{a}_{1s} \geq 0$ gelten. Daraus ergibt sich: $r \neq 1$. Also ist die erste Zeile nicht die Pivotzeile. Daraus folgt, dass nach dem k_0-ten Durchlauf des dualen Programms als Pivotspalten nur Spalten $\overline{A}_{\cdot j}$ in Frage kommen mit $\overline{a}_{0j} = 0$ und dass die erste Zeile $\overline{A}_{1\cdot}$ niemals Pivotzeile ist. Wird bei einer Pivotoperation der Wert $\overline{a}_{10}$ neu berechnet, so kommt wegen $r \neq 1$ die Formel 15.2.4. in Frage, also

$$\overline{\overline{a}}_{10} = \overline{a}_{10} - \frac{\overline{a}_{1s}\overline{a}_{r0}}{\overline{a}_{rs}}.$$

Aufgrund der Regel zur Auswahl von Pivotspalte und -zeile gilt $\overline{a}_{r0} < 0$, $\overline{a}_{rs} < 0$. Aus $\overline{a}_{1s} \geq 0$ folgt, dass $\overline{\overline{a}}_{10} \leq \overline{a}_{10}$ gilt. Mithin wird das Tableauelement $\overline{a}_{10}$ bei jeder Pivotoperation kleiner oder bleibt gleich. Würde $\overline{a}_{10}$ irgendwann einmal negativ, so könnte niemals mehr primale Zulässigkeit (d. h. $\overline{a}_{i0} \geq 0$, $i = 1, \ldots, m$) erreicht werden, da – wie wir oben gezeigt haben – die erste Zeile niemals Pivotzeile wird. Weil aber das duale Simplexverfahren nach endlich vielen Schritten aufhört, muss $\overline{a}_{10}$ immer nicht-negativ bleiben.

Daraus folgt, dass $\overline{a}_{10}$ nach dem k_0-ten Durchlauf des dualen Programms bei jeder dualen Pivotoperation nicht größer wird, aber nicht-negativ bleibt. Also ist die Folge $(\overline{a}^k_{10})_{k \geq k_0}$ monoton fallend und nach unten beschränkt.

Ist $\overline{a}_{10}$ in der Optimallösung eines dualen Programms nicht ganzzahlig, so wird aufgrund der Zusatzregel zu Schritt9 die Schnittebene aus der ersten Zeile abgeleitet. Ersetzen wir in der obigen Diskussion der Zielfunktion den Zeilenindex 0 durch den Zeilenindex 1, so erhalten wir analog, dass nach dem ersten anschließenden Pivotschritt gilt: $\overline{\overline{a}}_{10} \leq \lfloor \overline{a}_{10} \rfloor$. Wäre also $\overline{a}_{10}$ unendlich oft nicht ganzzahlig, so wäre die Folge $(\overline{a}^k_{10})_{k \geq k_0}$ nach unten unbeschränkt. Da aber 0 eine untere Schranke ist, muss $(a^k_{10})_{k \geq k_0}$ nach endlich vielen Schritten, sagen wir k_1, einen ganzzahligen Wert annehmen und diesen Wert von da an beibehalten.

Um zu zeigen, dass auch $\overline{a}_{20}$ nach endlich vielen weiteren Schritten ganzzahlig wird, können wir voraussetzen, dass $\overline{a}_{00}$ und $\overline{a}_{10}$ bereits ihren endgültigen ganzzahligen Wert angenommen haben. Dann verläuft der Beweis völlig analog wie bei $\overline{a}_{10}$. Auf diese Weise weisen wir nach, dass $\overline{a}_{00}, \overline{a}_{10}, \ldots, \overline{a}_{m0}$ nach endlich vielen Schritten einen ganzzahligen Wert annehmen. Damit ist der Beweis erledigt. □

Die Ganzzahligkeit aller Daten ist beim Gomory-Algorithmus unbedingt notwendig.

(7.32) Beispiel. Wir lösen das folgende IP mit dem ersten Gomory-Algorithmus:

$$\begin{aligned} \max\ & 3x_1 + x_2 & & \\ & 17x_1 + 11x_2 \leq 86.5 & & \\ & x_1 + 2x_2 \leq 10.2 & & \\ & x_1 \leq 3.87 & & \\ & x_1, x_2 \geq 0 & & \text{und ganzzahlig.} \end{aligned}$$

LP-Lösung:

		1	*2*
	0	−3	−1
3	86.5	17	11
4	10.2	1	2
5	3.87	$\boxed{1}$	0

$\longrightarrow$

		5	*2*
	11.61	3	−1
3	20.71	−17	$\boxed{11}$
4	6.33	−1	2
1	13.87	1	0

$\longrightarrow$

		5	*3*
	$^{14842}/_{1100}$	$^{16}/_{11}$	$^{1}/_{11}$
2	$-^{2071}/_{1100}$	$-^{17}/_{11}$	$^{1}/_{11}$
4	$-^{2821}/_{1100}$	$^{23}/_{11}$	$-^{2}/_{11}$
1	13.87	1	0
	$-^{543}/_{1100}$	$-^{5}/_{11}$	$\boxed{-\frac{1}{11}}$

$\longrightarrow$

		5	*6*
	13	1	1
2	$-^{1529}/_{1100}$	−2	1
4	$-^{3905}/_{1100}$	3	−2
1	3.87	1	0
3	5.42	5	−11
	$-f_{i0}$	0	0

Aus allen Zeilen des letzten Tableaus kann man einen Gomory-Schnitt ableiten, der die Form

$$0 \cdot x_5 + 0 \cdot x_6 \leq -f_{i0}$$

hat, mit $f_{i0} > 0$. Wäre dieser Schnitt gültig, hieße dies, dass unser Problem keine ganzzahlige Lösung hat, denn diese Ungleichung kann nicht erfüllt werden. Also liegt der Fehler in der Nichtganzzahligkeit der rechten Seite. △

Wir wissen ja bereits, dass das ganzzahlige Programmierungsproblem zur Klasse der $\mathcal{NP}$-vollständigen Probleme gehört. Man sollte also nicht erwarten, dass der Gomory-Algorithmus in polynomialer Zeit läuft. Selbst wenn wir statt des Simplexalgorithmus ein polynomiales Verfahren (Ellipsoidmethode, Karmarkar-Algorithmus) zur Lösung der jeweils auftretenden LPs wählen würden, wäre das gesamte Verfahren nicht polynomial, da die Zahl der Schnitte, die man betrachten muss, nicht durch ein Polynom beschränkt werden kann. Dies verdeutlicht das folgende ganzzahlige Programm.

(7.33) Beispiel. Für eine vorgegebene positive ganze Zahl k sei

$$\begin{aligned} \max\ & x_2 \\ -kx_1 + x_2 &\leq 0 \\ kx_1 + x_2 &\leq k \\ x_1, x_2 &\geq 0 \qquad \text{und ganzzahlig,} \end{aligned}$$

ein ganzzahliges Programm (vergleiche (7.22)(b)). Dieses Programm hat nur zwei ganzzahlige Lösungen, $\binom{1}{0}$ und $\binom{0}{0}$, die beide optimal sind.

Wendet man den Gomory-Algorithmus auf dieses IP an, so wird man feststellen, dass bis zur Auffindung einer Optimallösung $\lceil \frac{3k}{2} \rceil - 1$ Gomory-Schnitte benötigt werden. Die Inputlänge des obigen LP kann durch $8\langle k \rangle$ abgeschätzt werden. Die Anzahl der Gomory-Schnitte kann also nicht durch ein Polynom in $\log k$ beschränkt werden. Daraus folgt, dass die Anzahl der Iterationen des Gomory-Algorithmus nicht polynomial in den Ausgangsdaten ist. △

7.5 Ein Schnittebenenverfahren für gemischt-ganzzahlige Programme

Die vorhergehenden Überlegungen zur Lösung ganzzahliger Programme mit Hilfe von Schnittebenen-Verfahren lassen sich recht einfach auf den gemischt-ganzzahligen Fall übertragen. Hierzu müssen wir uns nur überlegen, wie wir eine geeignete Schnittebene ableiten können. Gegeben sei also ein gemischt-ganzzahliges Programm

$$\begin{aligned} \max\ & c^T x + d^T y \\ & Ax + By = b \\ & x, y \geq 0 \\ & x \text{ ganzzahlig.} \end{aligned}$$

Um Bezeichungen einfacher zu machen, wollen wir dieses Programm in folgender Form schreiben:

$$\begin{aligned} \max\ & c^T x \\ & Ax = b \\ & x \geq 0 \\ & x_i \text{ ganzzahlig}, \quad i = 1, \ldots, n_1 \leq n \end{aligned} \tag{MIP$^=$}$$

d. h. wir nehmen an, dass die Variablen $x_1, \ldots, x_{n_1}$ mit den niedrigen Indizes ganzzahlig sind und die übrigen rational. Wir setzen $N_1 := \{1, \ldots, n_1\}$.

Wir gehen wieder davon aus, dass wir die LP-Relaxierung (LMIP$^=$) von (MIP$^=$) betrachten und dort an einer Basislösung angelangt sind, für die wir eine Schnittebene ableiten wollen, falls die Variablen x_i, $i \in N_1$ nicht ganzzahlig sind.

(7.34) Lemma. *Gegeben sei ein gemischt-ganzzahliges Programm der Form* (MIP$^=$) *mit* $N_1 = \{1, \ldots, n_1\}$ *mit ganzzahligen Daten. Sei* A_B *eine Basis der LP-Relaxierung* (LMIP$^=$)*, dann gilt: Ist* $x_{B(i)}$ *eine Basisvariable mit* $B(i) \in N_1$ *und ist* $\overline{b}_i = \overline{a}_{i0} \notin \mathbb{Z}$*, so ist*

$$-\sum_{j \in N_{\mathbb{Z}}} f_{ij} x_j - \sum_{j \in N_{\mathbb{Q}}^+} \overline{a}_{ij} x_j + \sum_{j \in N_{\mathbb{Q}}^-} \frac{f_{i0}\overline{a}_{ij}}{1 - f_{i0}} x_j \leq -f_{i0} \tag{$*$}$$

eine Schnittebene, die die gegenwärtige Basislösung abschneidet, falls $f_{i0} = \overline{a}_{i0} - \lfloor \overline{a}_{i0} \rfloor = \overline{b}_i - \lfloor \overline{b}_i \rfloor \neq 0$ *gilt.*

Hierbei sei B *die Indexmenge der Basisvariablen,* N *die Indexmenge der Nichtbasisvariablen,* $N_{\mathbb{Z}}$ *die Indexmenge der Nichtbasisvariablen, die der Ganzzahligkeitsforderung unterliegen, und* $N_{\mathbb{Q}} := N \setminus N_{\mathbb{Z}}$*. Wie üblich ist* $f_{ij} = \overline{a}_{ij} - \lfloor \overline{a}_{ij} \rfloor$ *der gebrochene Teil, und es ist* $N_{\mathbb{Q}}^- := \{j \in N_{\mathbb{Q}} \mid \overline{a}_{ij} \leq 0\}$*,* $N_{\mathbb{Q}}^+ := \{j \in N_{\mathbb{Q}} \mid \overline{a}_{ij} > 0\}$*.* △

Beweis. O. B. d. A. sei $B = (1, 2, \ldots, m)$, $N = (m+1, \ldots, n)$. Wie üblich starten wir mit der Darstellung:

$$x_i = \overline{b}_i - \sum_{j \in N} \overline{a}_{ij} x_j = \overline{b}_i - \sum_{j \in N_{\mathbb{Q}}} \overline{a}_{ij} x_j - \sum_{j \in N_{\mathbb{Z}}} \overline{a}_{ij} x_j. \tag{1}$$

Daraus folgt

$$\sum_{j\in N_{\mathbb{Z}}} f_{ij}x_j + \sum_{j\in N_{\mathbb{Q}}} \overline{a}_{ij}x_j - f_{i0} = \lfloor \overline{a}_{i0} \rfloor - \sum_{j\in N_{\mathbb{Z}}} \lfloor \overline{a}_{ij} \rfloor x_j - x_i. \tag{2}$$

Die rechte Seite von (2) ist für alle zulässigen Lösungen von (MIP$^=$) ganzzahlig, also gilt entweder

$$\sum_{j\in N_{\mathbb{Z}}} f_{ij}x_j + \sum_{j\in N_{\mathbb{Q}}} \overline{a}_{ij}x_j - f_{i0} \geq 0 \tag{3}$$

oder

$$\sum_{j\in N_{\mathbb{Z}}} f_{ij}x_j + \sum_{j\in N_{\mathbb{Q}}} \overline{a}_{ij}x_j - f_{i0} \leq -1. \tag{4}$$

Falls (3) gilt, so gilt auch

$$\sum_{j\in N_{\mathbb{Z}}} f_{ij}x_j + \sum_{j\in N_{\mathbb{Q}}^+} \overline{a}_{ij}x_j \geq f_{i0}. \tag{5}$$

Falls (4) gilt, so gilt (da für alle j gilt: $f_{ij} \geq 0$):

$$\sum_{j\in N_{\mathbb{Q}}^-} \overline{a}_{ij}x_j \leq f_{i0} - 1. \tag{6}$$

Multiplizieren wir (6) mit $\frac{f_{i0}}{f_{i0}-1} < 0$, so erhalten wir

$$-\sum_{j\in N_{\mathbb{Q}}^-} \frac{f_{i0}\overline{a}_{ij}}{1-f_{i0}}x_j \geq f_{i0}. \tag{7}$$

Die linken Seiten von (5) und (7) sind beide nicht-negativ; da (5) oder (7) gelten muss, erhalten wir durch Aufaddieren der linken Seiten

$$\sum_{j\in N_{\mathbb{Z}}} f_{ij}x_j + \sum_{j\in N_{\mathbb{Q}}^+} \overline{a}_{ij}x_j - \sum_{j\in N_{\mathbb{Q}}^-} \frac{f_{i0}\overline{a}_{ij}}{1-f_{i0}}x_j \geq f_{i0}, \tag{8}$$

also ist die angegebene Ungleichung für alle zulässigen Lösungen gültig. Da für die gegenwärtige Basislösung $x_N = 0$ gilt, erfüllt diese Basislösung die Ungleichung nicht. □

Durch eine genaue Analyse des obigen Beweises kann man den Schnitt $(*)$ noch etwas verbessern.

(7.35) Lemma. *Die Voraussetzungen seien wie in Lemma (7.34). Falls es ein $j \in N_{\mathbb{Z}}$ gibt, mit $f_{ij} > f_{i0}$, so kann die obige Ungleichung zu folgender Ungleichung verschärft werden:*

$$-\sum_{j\in N_{\mathbb{Z}}^+} f_{ij}x_j - \sum_{j\in N_{\mathbb{Z}}^-} \frac{f_{i0}(1-f_{ij})}{1-f_{i0}}x_j - \sum_{j\in N_{\mathbb{Q}}^+} \overline{a}_{ij}x_j + \sum_{j\in N_{\mathbb{Q}}^-} \frac{f_{i0}\overline{a}_{ij}}{1-f_{i0}}x_j \leq -f_{i0}, \qquad (**)$$

wobei $N_{\mathbb{Z}}^+ = \{j \in N_{\mathbb{Z}} \mid f_{ij} \leq f_{i0}\}$, $N_{\mathbb{Z}}^- = \{j \in N_{\mathbb{Z}} \mid f_{ij} > f_{i0}\}$. △

Beweis. Da nach Annahme $f_{i0} > 0$, gilt $f_{ij} > 0 \; \forall j \in N_{\mathbb{Z}}^-$. Subtrahieren wir $\sum_{j \in N_{\mathbb{Z}}^-} x_j$ von beiden Seiten von (2), so erhalten wir:

$$\sum_{j \in N_{\mathbb{Z}}^+} f_{ij} x_j + \sum_{j \in N_{\mathbb{Z}}^-} (f_{ij} - 1) x_j + \sum_{j \in N_{\mathbb{Q}}} \overline{a}_{ij} x_j - f_{i0} = \lfloor \overline{a}_{i0} \rfloor - \sum_{j \in N_{\mathbb{Z}}^+} \lfloor \overline{a}_{ij} \rfloor x_j - \sum_{j \in N_{\mathbb{Z}}^-} \lceil \overline{a}_{ij} \rceil x_j - x_i. \tag{9}$$

Da die rechte Seite wiederum ganzzahlig ist, erhalten wir, dass entweder

$$\sum_{j \in N_{\mathbb{Z}}^+} f_{ij} x_j + \sum_{j \in N_{\mathbb{Q}}^+} \overline{a}_{ij} x_j \geq f_{i0} \tag{10}$$

oder

$$-\frac{f_{i0}}{1 - f_{i0}} \Big(\sum_{j \in N_{\mathbb{Z}}^-} (f_{ij} - 1) x_j + \sum_{j \in N_{\mathbb{Q}}^-} \overline{a}_{ij} x_j \Big) \geq f_{i0} \tag{11}$$

gilt.

Also muss die Summe der beiden linken Seiten größer oder gleich f_{i0} sein. Diese Summe ist gerade die behauptete Ungleichung.

Die Koeffizienten von $(*)$ sind die gleichen wie die von $(**)$, außer im Falle $j \in N_{\mathbb{Z}}^-$. Aus $f_{ij} > f_{i0}$ für $j \in N_{\mathbb{Z}}^-$ folgt

$$f_{ij} - f_{i0} f_{ij} > f_{i0} - f_{i0} f_{ij}$$

und damit

$$f_{ij} > \frac{f_{i0}(1 - f_{ij})}{1 - f_{i0}}.$$

Daraus folgt, dass $(**)$ schärfer ist als $(*)$. □

(7.36) Algorithmus (Gomory's gemischt-ganzzahliger Algorithmus).
Eingabe: $A \in \mathbb{Z}^{(m,n)}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$, Indexmenge N_1.
Ausgabe: Lösung des gemischt-ganzzahligen Programms (MIP$^=$)

$$\begin{aligned} \max \; & c^T x \\ & Ax = b \\ & x \geq 0 \\ & x_i \text{ ganzzahlig}, \quad i \in N_1 \end{aligned}$$

Setze wie in (7.27): $\overline{a}_{i0} := b_i$, $i = 1, \ldots, m$, $\overline{a}_{0j} := -c_j$, $j = 1, \ldots, n$, $\overline{a}_{00} :=$ Zielfunktionswert.

1. Löse das zugehörige lineare Programm (LMIP$^=$) mit dem primalen Simplexalgorithmus.
2. Gilt $\overline{a}_{i0} \in \mathbb{Z}$ für alle Basisvariablen $x_{B(i)}$ mit $B(i) \in N_1$, so ist eine optimale gemischt-ganzzahlige Lösung von (MIP$^=$) gefunden.

3. Wähle ein $i \in \{0, 1, \ldots, m\}$ mit $\overline{a}_{i0} \notin \mathbb{Z}$, das zu einer ganzzahligen Variablen gehört, und setze

$$f_{ij} := \overline{a}_{ij} - \lfloor \overline{a}_{ij} \rfloor \qquad \text{für alle} j \in N \text{ (Nichtbasisvariable) mit } j \in N_1$$
$$f_{00} := \overline{a}_{00} - \lfloor \overline{a}_{00} \rfloor$$
$$\overline{a}_{m+1,j} := \begin{cases} -f_{ij}, & \text{falls } j \in N_1 \text{ und } f_{ij} \leq f_{i0} \\ -\frac{f_{i0}(1-f_{ij})}{1-f_{i0}}, & \text{falls } j \in N_1 \text{ und } \quad f_{ij} > f_{i0} \\ -\overline{a}_{ij}, & \text{falls } j \in \{1, \ldots, n\} \setminus N_1 \text{ und } \overline{a}_{ij} > 0 \\ \frac{f_{i0}\overline{a}_{ij}}{1-f_{i0}}, & \text{falls } j \in \{1, \ldots, n\} \setminus N_1 \text{ und } \overline{a}_{ij} \leq 0 \end{cases}$$

Füge die Zeile

$$\overline{a}_{m+1,0}, \overline{a}_{m+1,1}, \ldots, \overline{a}_{m+1,n-m}$$

als $(m+1)$-te Zeile zum gegenwärtigen Tableau hinzu. Setze $m := m+1$, $n := n+1$ und erweitere die Basis um die Schlupfvariable $n+1$.

4. Löse das erweiterte Programm mit dem dualen Simplexverfahren und gehe anschließend zu 2. △

Für das oben angegebene Verfahren für gemischt-ganzzahlige Programme kann man nun analog zu Satz (7.31) zeigen:

(7.37) Satz. *Falls die Zusatzregeln (7.30) zum ersten Gomory-Verfahren analog beim gemischt-ganzzahligen Gomory-Verfahren angewendet werden und unter der zusätzlichen Voraussetzung, dass der Wert der Zielfunktion in der Optimallösung ganzzahlig sein muss, dann liefert der gemischt-ganzzahlige Algorithmus von Gomory nach endlich vielen Schritten eine Optimallösung oder zeigt nach endlich vielen Schritten an, dass entweder* $(\mathrm{MIP}^{=})$ *unbeschränkt oder* $\mathrm{MIP}^{=}(A, B, b)$ *leer ist.* △

Beweis. Völlig analog zu (7.31). □

Die beiden Schnittebenentypen, die wir bisher kennengelernt haben, sind bei weitem nicht alle, die in Schnittebenenverfahren eingesetzt werden. Die sechziger Jahre brachten eine Fülle interessanter Studien zu Schnittebenenverfahren. Die neu erfundenen Schnittebenen erhielten schöne Namen wie

- Intersection-Cuts,
- Diamond-Cuts,
- Polaroid-Cuts,
- Cuts from the Hypercube,
- Outer-Polar-Cuts

(siehe hierzu Garfinkel und Nemhauser (1972)), jedoch änderten die neuen Techniken nichts an der bis Mitte der 90er Jahre empirisch beobachteten Tatsache, dass Schnittebenenverfahren dieser allgemeinen Art relativ ineffiziente Methoden zur Lösung ganzzahliger und gemischt-ganzzahliger Programme sind. Neuimplementierungen von Schnittebenenverfahren in kommerziellen MIP-Codes Ende der 90er Jahre haben zu einer Änderung der „herrschenden Meinung“ geführt. Die (praktisch effiziente) Einbindung von Schnittebenen in Branch & Bound-Codes hat zu einer enormen Beschleunigung dieser Verfahren geführt, siehe Bixby (2002).

Es ist natürlich nicht bei den oben genannten Schnitten geblieben. Seit sich Schnittebenenverfahren auch in der Praxis zur Lösung von MIPs durchgesetzt haben, sind alte Ideen neu aufgenommen und neue Schnitterzeugungsmethoden erfunden worden. Hier sind u. a. Mixed-Integer Rounding, Split Cuts, Cuts form Disjunctions, Lift and Project zu nennen, sowie grundsätzliche Untersuchungen zu „Cut Generating Functions“. In dem Buch Conforti et al. (2014) und dem Artikel Marchand et al. (2000) werden einige dieser Schnitte dargestellt und ihre Beziehungen untereinander erläutert. Schnittebenenentwurf und -erkennung ist ein aktives und aktuelles Forschungsgebiet, das jetzt zusätzlich noch die Lösung von nichtlinearen gemischt-ganzzahligen Optimierungsproblemen aufgreift. Auch hier sind bereits einige praktische Erfolge erzielt worden.

Die beiden Verfahren, die wir hier behandelt haben, werden auch zu den dualen Verfahren gezählt und zwar, weil in jedem Schritt eine dual zulässige Lösung behalten wird und das Ziel ist, eine primal zulässige und somit dann eine optimale Lösung zu finden.

Da unsere Tableaus auch gebrochene Werte enthalten und aus den gebrochenen rechten Seiten und den zugehörigen Tableau-Zeilen Schnitte abgeleitet werden, nennt man die Verfahren (7.27) bzw. (7.36) in der Literatur auch

Dual Fractional Integer Programming Algorithm
bzw.
Dual Fractional Mixed Integer Programming Algorithm.

Es gibt zu diesem Ansatz einige Varianten, und zwar kann man versuchen, mit einer primal zulässigen Lösung zu starten und immer dann, wenn im nächsten Simplexschritt eine nicht-ganzzahlige Lösung generiert wird, einen Schritt hinzuzufügen und auf diese Weise immer primale Zulässigkeit zu erhalten. Am Ende folgt dann aus der dualen Zulässigkeit die Optimalität. Verfahren dieses Typs nennt man *primale Schnittebenenverfahren.*

Ein großes Problem, das bei den „Fractional-Verfahren“ auftritt, sind die Rundefehler, die häufig zum Abbruch ohne Optimalität führen. Um diese unangenehmen Erscheinungen zu vermeiden, hat Gomory ein Verfahren entwickelt, das mit rein ganzzahliger Arithmetik arbeitet und deshalb *All-Integer Verfahren* genannt wird. Hierzu sind ebenfalls Varianten angegeben worden, so dass wir *Dual-All-Integer* und *Primal-All-Integer Integer Programming Verfahren* kennen. Aus Zeitgründen können wir auf diese Varianten nicht eingehen. In der Praxis werden sie nicht verwendet.

Praxis

Die Gomory-Schnitte werden natürlich in allen kommerziellen und guten akademischen Codes zur Lösung von ganzzahligen und gemischt-ganzzahligen Optimierungsproblemen verwendet. Sie alleine führen jedoch nicht zum Erfolg. Jeder gute Code enthält heute eine Vielzahl von zusätzlichen Separatoren, das sind Unterprogramme, die nach Schnittebenen einer jeweils speziellen Form suchen. Mehrere dieser Separatoren sind inspiriert durch polyedertheoretische Überlegungen, wie wir sie in Kapitel 3 vorgenommen haben. So haben insbesondere Facetten des Knapsack-Polyeders, von verschiedenen Fluss-Polyedern (Fow-Cover-Ungleichungen) und des Stabile-Mengen-Polyeders (Cliquen- und Ungerade-Kreis-Ungleichungen) Eingang in MIP-Codes gefunden. Praktische Erfahrung mit MIPs hat gezeigt, dass einige Klassen dieser speziellen Schnittebenen häufig bei der Lösung von allgemeinen MIPs sehr hilfreich sein können. Die Auswahl der jeweils noch zusätzlich betrachteten Schnittebenen ist im Wesentlichen erfahrungsbasiert. Welche Separatoren wann und unter welchen Umständen eingesetzt werden, wird durch umfangeiche Testläufe entschieden. Theoretische Konvergenzüberlegungen spielen dabei keine Rolle. Viel wichtiger sind Fortschritte in der Zielfunktion (tiefe Schnitte) und numerische Stabilität. Zu diesem Thema kann man eine eigene Vorlesung gestalten zumal dann, wenn man auch noch die Kombination von Schnittebenenverfahren mit Branch & Bound (und Column Generation) behandeln will, denn hierbei ergeben sich weitere praktisch wichtige Probleme. Soll man in den Knoten eines Branch & Bound-Baumes weiterhin Schnittebenen erzeugen. Diese sind u. U. nicht global gültig. Ist es sinnvoll, das trotzdem zu tun, etc.

Kommerzielle Codes geben nur wenige ihrer „Geheimnisse“preis. Über den Code SCIP, der am ZIB entwickelt wurde und weiter entwickelt wird, kann man sich relativ umfassend informieren, siehe `http://scip.zib.de/`. Wie man SCIP verwendet, selbst erweitern und einsetzen kann, ist auf der Seite `http://scip.zib.de/doc/html/` beschrieben. Die Webseite `http://scip.zib.de/doc/html/group__SEPARATORS.php` listet zum Beispiel die 13 Separatoren auf, die derzeit in SCIP implementiert sind. Wie man einen eigenen Separator zu SCIP hinzufügen kann, wird auf `http://scip.zib.de/doc/html/SEPA.php` erklärt. Moderne MIP-Codes bestehen aus mehreren hunderttausend Lines, sind also höchst komplex und erfordern sorgfältiges Entwicklungsmanagement. Praktische Effizienz wird nur durch gute Kenntnisse über das Zusammenspiel aller Code-Komponenten erreicht, welche nur durch zeitaufwändige Experimente mit großen, realen Datensätzen erworben werden können.

Literaturverzeichnis

D. Bertsimas und R. Weismantel. *Optimization over Integers.* Dynamic Ideas, Belmont, Massachusetts, 2005.

R. E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50:3–15, 2002.

V. Chvátal. Edmonds' polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973.

M. Conforti, G. Cornuejols, und G. Zambelli. *Integer Programming.* Springer, 2014.

R. S. Garfinkel und G. L. Nemhauser. *Integer programming.* Series in Decision and Control. Wiley, New York, 1972.

H. Marchand, A. Martin, R. Weismantel, und L. Wolsey. Cutting plans in integer and mixed integer programming. *Discrete Applied Mathematics*, 123:397–446, 2000.

G. L. Nemhauser und L. A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, 1999.

A. Schrijver. On cutting planes. *Annals of Discrete Mathematics*, 9:291–296, 1980.

8 Heuristiken

8.1 Primale Heuristiken für schwere Probleme: Eröffnungs- und Verbesserungsverfahren

Es ist leider so, dass viele der in der Praxis vorkommenden kombinatorischen Optimierungsprobleme groß und — im Sinne der Komplexitätstheorie — schwer sind. Ferner müssen viele dieser Probleme häufig in beschränkter Zeit mit nicht allzu großen Computern „gelöst“ werden. Diesen Praxisanforderungen kann man mit zwei möglichen Ansätzen entsprechen. Entweder man entwickelt — falls das möglich ist — Verfahren, die für die vorliegenden speziellen Probleme exakt arbeiten (also Optimallösungen liefern) und empirisch für die gegebenen Größenordnungen vertretbaren Rechenaufwand erfordern, oder man entwirft approximative Verfahren (Heuristiken), die in kurzer Zeit obere und untere Schranken für den Optimalwert liefern und somit gewisse Güteaussagen erlauben.

Für spezielle Anwendungsprobleme mit problemspezifischen Nebenbedingungen können natürlich individuelle Heuristiken entwickelt werden, die auf die besonderen Nebenbedingungen abgestellt sind. Es gibt jedoch einige Prinzipien, die (mit geeigneten Modifikationen) bei allen schwierigen kombinatorischen Optimierungsproblemen zur Entwicklung von Heuristiken eingesetzt werden können. Diese Prinzipien sollen in diesem Kapitel erläutert und (meistens) anhand des symmetrischen Travelling Salesman Problems eingehend erklärt werden.

Wir wollen solche Heuristiken, die zulässige Lösungen des vorliegenden Optimierungsproblems liefern, **primale Heuristiken** nennen. Es gibt unzählige Versuche, Heuristiken zu klassifizieren. Wir wollen diese Versuche hier nicht weiter kommentieren und werten. Im weiteren werden wir primale Heuristiken in zwei Verfahrensklassen aufteilen, also nur eine ganz grobe Gliederung vornehmen. Die beiden Klassen wollen wir mit Eröffnungsverfahren (oder Startheuristiken) und mit Verbesserungsverfahren bezeichnen.

Unter **Eröffnungsverfahren** versteht man solche Algorithmen für kombinatorische Optimierungsprobleme, die mit der leeren Menge (oder einer „trivialen Anfangsmenge“) beginnend sukzessive eine zulässige Lösung aufbauen, wobei beim Aufbau **lokale Optimierungsüberlegungen** angestellt werden. Ein typisches Beispiel hierfür ist der uns bereits bekannte Greedy-Algorithmus, bei dem man jeweils zur gegenwärtigen (Teil-)Lösung I dasjenige Element e der Grundmenge zu I hinzufügt, das unter den noch nicht berücksichtigten Elementen den besten Zielfunktionswert hat, wobei das Hinzufügen von e nicht zur Unzulässigkeit führen darf.

Verbesserungsverfahren sind solche Algorithmen, die mit einer zulässigen Anfangslösung beginnen und wiederum unter Berücksichtigung lokaler Optimalitätsbedingungen gewisse Änderungen an der bisherigen Lösung vornehmen und so fortschreitend zu einer

„lokalen Optimallösung“ gelangen. Hier gibt es viele Variationsmöglichkeiten. Man kann Verbesserungsverfahren auf mehrere Startlösungen parallel oder sukzessiv anwenden. Um das Verbleiben in „lokalen Optima“ möglichst zu verhindern, kann man ab und zu zufällige Veränderungen (und sogar Verschlechterungen) zulassen. Man muss nicht immer mit den gleichen „Verbesserungstricks“ arbeiten, gelegentliche Variation der Modifikationstechniken kann helfen, etc. Was man wie im Einzelnen macht, hängt natürlich ab von den Erfahrungsberichten anderer, von speziellen Eigenschaften des betrachteten Systems, der verfügbaren Rechenzeit und dem Speicherplatzbedarf.

In der Regel werden Heuristiken eingesetzt, um bei schweren Problemen in relativ kurzer Zeit einigermaßen gute Lösungen zu finden. Wenn vorliegende Problembeispiele sehr groß und exakte Verfahren nicht vorhanden sind, muss man mit Heuristiken vorlieb nehmen und versuchen, im vorgegebenen Rechner- und Zeitrahmen „das Beste“ zu erreichen.

8.2 Eröffnungsheuristiken für symmetrisches TSP

Wir erinnern daran, dass das symmetrische Travelling Salesman Problem (TSP) die Aufgabe ist, in einem ungerichteten vollständigen Graphen $K_n = (V, E)$ mit Kantengewichten einen hamiltonschen Kreis (auch *Tour* genannt) zu finden, der möglichst geringes Gewicht hat. Das TSP ist das vermutlich am besten untersuchte kombinatorische Optimierungsproblem. Die Anzahl der Veröffentlichungen ist fast unübersehbar. Das TSP ist auch eine beliebte Spielwiese für Amateure (dazu gehören auch Wissenschaftler aus der Physik, den Ingenieurwissenschaften, der Biologie und einigen anderen Bereichen), die neue (oder bekannte, aber mit neuem Namen versehene) Heuristiken erfinden, häufig auf Analogien zu physikalischem oder biologischem Vorgehen fußend, und die nicht selten kühne Behauptungen über die Leistungsfähigkeit dieser Verfahren aufstellen. Manches davon findet den Weg in die Tagespresse.

Ein sehr gutes Buch zum TSP ist der Sammelband Lawler et al. (1985), welcher – obwohl schon einige Jahre alt – ausgezeichnete Information zum TSP enthält. Ein neuerer Sammelband zum TSP mit Ergänzungen zu Lawler et al. (1985) ist das Buch Gutin und Punnen (2002).

Das Buch Reinelt (1994) enthält eine sorgfältige Studie für Heuristiken zur Bestimmung von unteren und oberen Schranken für den Wert einer optimalen Tour. Die zur Analyse der verschiedenen Verfahren benutzten TSP-Beispiele (fast alle aus der Praxis) sind von G. Reinelt elektronisch verfügbar gemacht worden. Diese Sammlung, genannt *TSPLIB*, ist inzwischen durch Hinzunahme sehr großer Problembeispiele erheblich erweitert worden, sie ist aufrufbar unter der Webadresse

`https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95`

Sehr gute Informationen zum TSP (u. a. Bilder der „Weltrekorde“, d. h. der größten exakt gelösten TSPs) findet man auf der von D. Applegate, B. Bixby, V. Chvátal und B. Cook angelegten und von Bill Cook gepflegten Webpage, siehe

`http://www.math.uwaterloo.ca/tsp`

Dieses Autorenteam hat derzeit den besten Code zur exakten Lösung großer TSPs. Es handelt sich hierbei um ein LP-basiertes Verfahren mit Schnittebenen und Branch&Bound, in das viele Heuristiken eingeflossen sind. Den „Concorde TSP Solver“ kann man sich von der o. g. Webpage herunterladen.

Das Buch Applegate et al. (2006) basiert auf den Erfahrungen, die die vier Autoren bei der Entwicklung des TSP-Codes Concorde gemacht haben. Es enthält eine umfangreiche Literatursammlung, viele Anwendungsbeispiele und einen Überblick über die historische Entwicklung der TSP-Lösungsverfahren, ein sehr empfehlenswertes Buch! Eine weitere interessante Homepage zum TSP ist TSPBIB:

`http://frcatel.fri.utc.sk/users/pesko/Studenti/TSPBIB.htm`

mit Verweisen zu vielen anderen elektronischen Quellen zum TSP, u. a. zu Seiten, wo man die Ausführung einiger Heuristiken graphisch verfolgen kann.

Der Aufsatz Grötschel (2007), Kapitel 4 eines Buches, das sich an Lehrer richtet, gibt eine kurze Einführung in das TSP, der Artikel Grötschel und Padberg (1999) ist analog für einen breiten Leserkreis geschrieben.

Hamiltonsche Kreise.

Bevor wir uns dem TSP zuwenden, wollen wir kurz der Frage nachgehen, unter welchen Bedingungen an einen Graphen auf die Existenz eines hamiltonschen Kreises geschlossen werden kann. Da das Problem $\mathcal{NP}$-vollständig ist, kann man natürlich keine vollständige Lösung des Problems erwarten. Alle hinreichenden Bedingungen fordern im Prinzip, dass ein Graph viele, gut verteilte Kanten haben muss, wenn er hamiltonsch sein soll. Eine einfache Idee ist, für jeden Knoten zu fordern, dass er einen hohen Grad hat. Die Idee funktioniert jedoch nicht direkt. Man überlegt sich leicht, dass es zu jedem $d \in \mathbb{N}$ einen Graphen G mit Minimalgrad $\delta(G) \geq d$ gibt, der keinen hamiltonschen Kreis besitzt. Ein Klassiker des Gebiets ist die folgende Beobachtung von Dirac aus dem Jahre 1952.

(8.1) Satz. *Ist $G = (V, E)$ ein einfacher Graph mit $n = |V| \geq 3$ und Minimalgrad $\delta(G) \geq \frac{n}{2}$, so enthält G einen Hamiltonkreis.* △

Beweis. Sei $P = v_1, \ldots, v_k$ ein längster Weg in G. Hätte v_1 einen Nachbarn außerhalb von P, könnte der Weg verlängert werden, was wegen seiner Maximalität nicht geht. Analog liegen auch alle Nachbarn von v_k in P. Mindestens $\frac{n}{2}$ der höchsten $n-1$ Knoten $v_1, \ldots, v_{k-1}$ sind Vorgänger auf P eines Nachbarn von v_1, und mindestens $\frac{n}{2}$ der Knoten $v_1, \ldots, v_{k-1}$ sind Nachbarn von v_k. Folglich muss es einen Knoten, sagen wir v_i, $1 \leq i < k$, geben mit $v_1 v_{i+1} \in E$ und $v_i v_k \in E$. Damit ist $C = v_1 v_{i+1} P v_k v_i P v_1$ ein Kreis in G. Gäbe es einen Knoten $v_0 \in V$, der nicht in C ist, so gäbe es (wegen $\deg(v_0) \geq \frac{n}{2}$) zwei Nachbarn von v_0, die auf C benachbart sind. Wir könnten also C verlängern und hätten damit auch einen Weg gefunden, der länger als P ist, Widerspruch. Also ist C ein hamiltonscher Kreis. □

Die hinreichende Bedingung des Satzes von Dirac ist sukzessive von verschiedenen Autoren verfeinert worden. Wir geben hier zwei Resultate dieser Art an. Die Beweise

verlaufen ähnlich, die Konstruktionen sind natürlich komplizierter. Das nächste Resultat stammt von Chvátal und ist aus dem Jahr 1972.

Eine Folge $d_1, d_2, \ldots, d_n$ nennen wir *Gradsequenz*, wenn es einen einfachen Graphen G mit n Knoten $1, \ldots, n$ gibt, so dass $\deg(i) = d_i$, $i = 1, \ldots, n$. Wir nennen eine Gradsequenz *hamiltonsch*, wenn jeder Graph mit dieser Gradsequenz hamiltonsch ist.

(8.2) Satz. *Eine Gradsequenz $d_1, d_2, \ldots, d_n$ ist genau dann hamiltonsch, wenn für jedes i, $1 \leq i < \frac{n}{2}$ gilt:*

$$d_i \leq i \implies d_{n-i} \geq n - i.$$

△

Bondy und Chvátal beobachteten 1974, dass sich der Diracsche Beweis so modifizieren lässt, dass man folgende (stärkere) Bedingung erhält.

(8.3) Satz. *Sei G ein einfacher Graph mit n Knoten, und seien $u, v \in V$ zwei nichtadjazente Knoten, so dasss*

$$\deg(u) + \deg(v) \geq n.$$

Dann gilt: G ist hamiltonsch genau dann, wenn $G + uv$ hamiltonsch ist. △

Sei G ein beliebiger Graph mit n Knoten. Der *Abschluss* von G ist derjenige Graph, den man aus G dadurch erhält, dass man rekursiv für jedes Paar u, v nichtadjazenter Knoten mit Gradsumme $\geq n$ die Kante uv zu G hinzufügt. Man kann leicht zeigen, dass der Abschluss eindeutig bestimmt (also unabhängig von der Reihenfolge des Hinzufügens) ist. Satz (8.3) liefert dann durch Rekursion:

(8.4) Satz. *Ein einfacher Graph G ist genau dann hamiltonsch, wenn sein Abschluss hamiltonsch ist.* △

Ist insbesondere der Abschluss von G der vollständige Graph K_n, so kann man daraus schließen, dass G hamiltonsch ist.

Die Bedingungen der oben angegebenen Sätze implizieren, dass jeder Graph mit n Knoten, der eine solche Bedingung erfüllt, $O(n^2)$ Kanten enthält. Die Theorie der Zufallsgraphen liefert schärfere Resultate. Man kann zeigen, dass ein zufälliger Graph mit $O(n \log n)$ Kanten mit hoher Wahrscheinlichkeit hamiltonsch ist.

Warum werden Sätze vom Typ (8.1), (8.2) oder (8.3) in einem Kapitel über Heuristiken betrachtet? Ein Grund dafür (neben dem Wunsch des Vortragenden, ein paar interessante Resultate der Graphentheorie zu vermitteln) ist, dass die Beweise der Sätze algorithmisch sind. In der Tat sind sehr viele Beweise in der Graphentheorie algorithmisch; sie werden häufig nur nicht so formuliert, weil die Autoren entweder möglichst kurze Beweise geben wollen oder an Algorithmen nicht wirklich interessiert sind.

Schauen wir uns also den Beweis des Satzes von Dirac genauer an und interpretieren wir ihn als Heuristik.

Wir wählen einen beliebigen Knoten von $G = (V, E)$, sagen wir v; dann gehen wir zu einem Nachbarn von v, sagen wir v'; von v' aus besuchen wir einen Nachbarn, den wir bisher nicht besucht haben. Wir führen diese Depth-First-Suche solange durch, bis wir zu einem Knoten kommen, dessen Nachbarn alle bereits in dem bisher konstruierten

Weg sind. Dann gehen wir zurück zu v, wählen einen weiteren (noch nicht besuchten) Nachbarn von v und verlängern den Weg so lange wie möglich. Wir erhalten auf diese Weise einen Weg $P = v_1, \ldots, v_k$, so dass alle Nachbarn von v_1 und von v_k in P liegen. Dies ist eine Trivialheuristik zur Konstruktion eines maximalen (nicht notwendigerweise längsten) Weges in G.

Nun gehen wir genau wie im Beweis von Satz (8.1) vor. Aufgrund der Bedingung $\delta(G) \geq \frac{n}{2}$ können wir einen Knoten v_i, $1 \leq i < k$, finden mit $v_1, v_{i+1} \in E$ und $v_i, v_k \in E$. Wir schließen P zu einem Kreis $C = v_1 v_{i+1} P v_k v_i P v_1$.

Gibt es einen Knoten $v_0 \in V$, der nicht in C ist, so hat v_0 (wegen $\deg(v_0) \geq \frac{n}{2}$) zwei Nachbarn in C, die auf C benachbart sind, sagen wir v_j und v_{j+1}. Wir entfernen die Kante $v_j v_{j+1}$ aus C und fügen die beiden Kanten $v_j v_0$ und $v_0 v_{j+1}$ ein, verlängern also C auf diese Weise. Gibt es einen weiteren Knoten außerhalb des neuen Kreises, so wiederholen wir diese Prozedur. Gibt es keinen Knoten mehr, der nicht in C liegt, so haben wir einen hamiltonschen Kreis gefunden.

Den Satz von Dirac kann man auch als Algorithmusanalyse interpretieren. Wir erfinden eine Heuristik (erst langen Weg konstruieren, diesen zum Kreis schließen, alle restlichen Knoten einbauen) und formulieren dann eine Bedingung (hier $\delta(G) \geq \frac{n}{2}$), so dass die Heuristik beweisbar einen hamiltonschen Kreis liefert. Es ist recht instruktiv, graphentheoretische Sätze einmal auf die oben beschriebene Weise zu durchleuchten. Sehr häufig sind die Resultate und Beweise auf genau die hier beschriebene Weise interpretierbar.

TSP-Eröffnungsverfahren

Die im nachfolgenden beschriebenen TSP-Heuristiken folgen übrigens in verschiedener Hinsicht den gleichen Prinzipien, die oben dargelegt wurden. Bei der Auswahl der Ausbaustrategie (Verlängerung des bestehenden Weges oder Kreises) werden Auswahlregeln benutzt, die in einem spezifizierbaren Sinn „greedy“ sind. Man versucht bei einer Eröffnungsheuristik, den nächsten Schritt so auszuführen, dass er bezüglich eines lokalen Kriteriums optimal ist. Natürlich muss gewährleistet sein, dass am Ende eine zulässige Lösung (ein hamiltonscher Kreis oder kurz Tour) gefunden wird. Der Kern aller derartigen Verfahren ist die „geschickte“Wahl des lokalen Optimalitätskriteriums. Es muss algorithmisch einfach sein (aus Rechenzeitgründen), die lokalen Optima zu bestimmen. Zum anderen soll gewährleistet sein, dass das lokale Kriterium zu einer „guten“ Lösung im Sinne der globalen Zielfunktion führt. Beim Greedy-Algorithmus bestand die Strategie darin, lokal das bezüglich des Gewichtes bestmögliche Element zu wählen, und wie Satz (2.14) zeigt, fährt man damit bei Maximierungsproblemen nicht allzu schlecht. Bei Minimierungsaufgaben ließ sich jedoch keine Gütegarantie finden.

(8.5) Algorithmus (Eröffnungsverfahren für das symmetrische TSP). Gegeben sei ein vollständiger Graph $K_n = (V, E)$, $n \geq 3$, mit „Kantenlängen“ c_{ij} für alle $ij \in E$.

(a) **Nächster Nachbar (NN)**

1. Wähle irgendeinen Knoten $i \in V$, markiere i, und setze $T := \emptyset$, $p := i$.
2. Sind alle Knoten markiert, setze $T := T \cup \{ip\}$ und STOP (T ist eine Tour).

3. Bestimme einen unmarkierten Knoten j, so dass

$$c_{pj} = \min\{c_{pk} \mid k \text{ unmarkiert}\}.$$

4. Setze $T := T \cup \{pj\}$, markiere j, setze $p := j$ und gehe zu 2.

(b) **Nearest Insert (NI)**

1. Wähle irgendeinen Kreis C der Länge drei.
2. Ist $V \setminus V(C) = \emptyset$, STOP; C ist eine Tour.
3. Bestimme einen Knoten $p \in V \setminus V(C)$, so dass ein Knoten $q \in V(C)$ existiert mit

$$c_{pq} = \min\{\min\{c_{ij} \mid j \in V(C)\} \mid i \in V \setminus V(C)\}.$$

4. Bestimme eine Kante $uv \in C$ mit

$$c_{up} + c_{pv} - c_{uv} = \min\{c_{ip} + c_{pj} - c_{ij} \mid ij \in C\}.$$

5. Setze $C := (C \setminus \{uv\}) \cup \{up, pv\}$ und gehe zu 2

(c) **Farthest Insert (FI)**
Wie NI, es wird lediglich Schritt 3 ersetzt durch

3. Bestimme einen Knoten $p \in V \setminus V(C)$, so dass ein Knoten $q \in V(C)$ existiert mit

$$c_{pq} = \max\{\min\{c_{ij} \mid j \in V(C)\} \mid i \in V \setminus V(C)\}.$$

(d) **Cheapest Insert (CI)** Wie NI, es werden lediglich die Schritte 3 und 4 ersetzt durch

3. Bestimme einen Knoten $p \in V \setminus V(C)$ und eine Kante $uv \in C$ mit

$$c_{up} + c_{pv} - c_{uv} = \min\{c_{ik} + c_{kj} - c_{ij} \mid ij \in C, k \in V \setminus V(C)\}.$$

(e) **Spanning-Tree-Heuristik (ST)**

1. Bestimme einen minimalen aufspannenden Baum B von K_n.
2. Verdopple alle Kanten aus B, um einen Graphen (V, B_2) zu erhalten.
3. (Da jeder Knoten in (V, B_2) einen geraden Grad hat und (V, B_2) zusammenhängend ist, enthält (V, B_2) eine Eulertour.) Bestimme eine Eulertour C, gib ihr eine Orientierung, wähle einen Knoten $i \in V$, markiere i, und setze $p := i$, $T := \emptyset$.
4. Sind alle Knoten markiert, setze $T := T \cup \{ip\}$ und STOP (T ist eine Tour).
5. Laufe von p entlang der Orientierung von C, bis ein unmarkierter Knoten, sagen wir q, erreicht ist. Setze $T := T \cup \{p, q\}$, markiere q, setze $p := q$ und gehe zu 4.

(f) **CHristofides-Heuristik (CH)**
Wie (ST), lediglich Schritt 2 wird ersetzt durch

	A	B	D	F	K	W
Aachen	–	91	80	259	70	121
Bonn	91	–	77	175	27	84
Düsseldorf	80	77	–	232	47	29
Frankfurt	259	175	232	–	189	236
Köln	70	27	47	189	–	55
Wuppertal	121	84	29	236	55	–

Tabelle 8.1: Entfernungen zwischen 6 Städten im Rheinland

2. Sei W die Menge der Knoten in (V, B) mit ungeradem Grad. (Man beachte: $|W|$ ist gerade!)
   a) Bestimme im von W induzierten Untergraphen von K_n ein perfektes Matching M minimalen Gewichts.
   b) Setze $B_2 := B \dot{\cup} M$ (Achtung: hier können Kanten doppelt vorkommen!) △

Wir wollen nun an einigen Beispielen die Wirkungsweisen der 7 oben angegebenen Heuristiken vorführen.

(8.6) Beispiel. Wir betrachten das durch Tabelle 8.1 definierte 6-Städte Problem (das *Rheinland-Problem*). Die Daten sind Straßenkilometer und entstammen einem Straßenatlas. Die geographische Lage der sechs Orte ist in Abb. 8.1 ungefähr maßstabgetreu angedeutet.

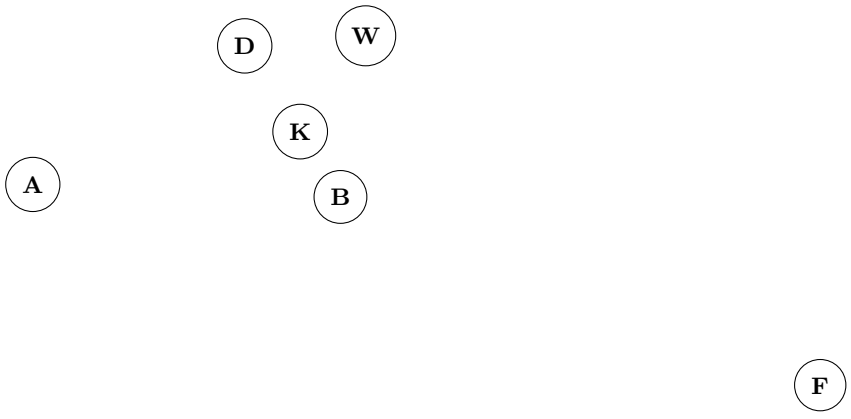


Abbildung 8.1: Ungefähre Lage von 6 Städten im Rheinland

(a) Wir wenden NN an

(a_1) Startpunkt A: Es ergibt sich die Rundreise

A — K — B — D — W — F — A

mit der Länge 698 km.

(a_2) Startpunkt F: Wir erhalten die Tour

F — B — K — D — W — A — F

der Länge 658 km.

(b) Wir wenden NI mit dem Startkreis K — D — B — K an. Dabei werden sukzessive die folgenden Kreise erzeugt:

K — D — B — K, K — D — W — B — K,
K — A — D — W — B — K, K — A — D — W — F — B — K.

Die so entstandene Tour hat die Länge 617 km.

(c) Wir wenden FI mit dem Startkreis K — A — F — K an. Wir erhalten sukzessiv die folgenden Kreise:

K — A — F — K, K — A — F — W — K,
K — A — F — W — D — K, K — A — B — F — W — D — K.

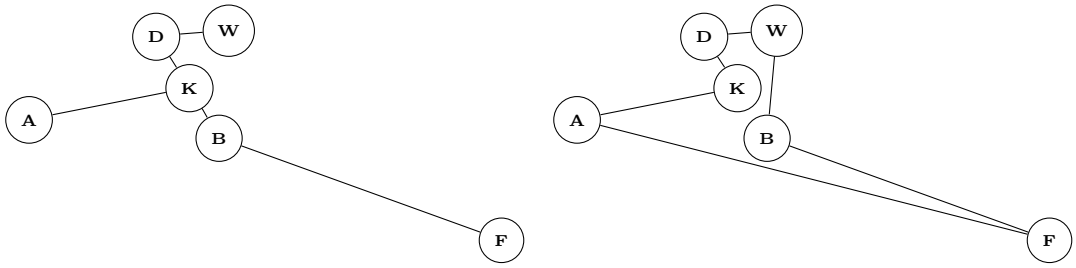
Die durch FI gefundene Tour hat die Länge 648 km.

(d) Mit CI und dem Startkreis A — B — F — A erhalten wir

A — B — F — A, A — K — B — F — A,
A — D — K — B — F — A, A — D — W — K — B — F — A.

Die so gefundene Tour hat die Länge 625 km.

(e) Der minimale aufspannende Baum B des Rheinland-Problems ist in Abbildung 8.2(a) gezeigt. Er hat die Länge 348 km. Wir verdoppeln die Kanten von B, wählen die



(a) Aufspannender Baum (b) Rundreise

Abbildung 8.2: ST auf dem Rheinland-Problem

folgende orientierte Eulertour AK, KD, DW, WD, DK, KB, BF, FB, BK, KA und starten mit A. Daraus ergibt sich die in Abbildung 8.2(b) gezeigte Rundreise der Länge 664 km.

(f) Im minimalen aufspannenden Baum B (siehe Abbildung 8.2(a)) haben die Knoten A, K, W und F ungeraden Grad. Das minimale perfekte Matching des durch {A, K, W, F} induzierten Untergraphen besteht aus den Kanten $M = \{\mathrm{AK}, \mathrm{WF}\}$. Sei $B_2 = B \cup M$. Wir wählen die orientierte Eulertour AK, KD, DW, WF, FB, BK, KA von (V, B_2), starten in A und erhalten die Tour A — K — D — W — F — B — A der Länge 648 km. Starten wir dagegen in B, so ergibt sich die Tour B — K — A — D — W — F — B der Länge 617 km. △

Die durch unsere Heuristiken gefundene kürzeste Rundreise hat also die Länge 617 km und ist in Abbildung 8.3 gezeigt.

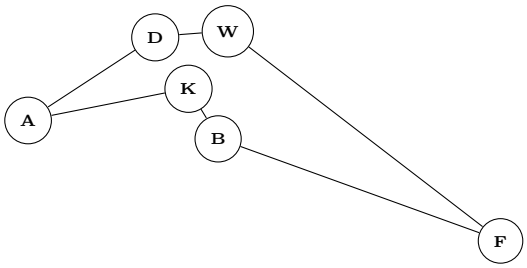


Abbildung 8.3: Kürzeste Rundreise durch 6 Städte im Rheinland

Beispiel (8.6) zeigt deutlich, dass die Heuristiken durchaus unterschiedliche Ergebnisse liefern können. Selbst einunddieselbe Heuristik kann bei verschiedener Wahl der noch offenen Parameter (z. B. Startpunkt) zu stark voneinander abweichenden Lösungen führen.

Wir wollen nun untersuchen, ob man theoretisch etwas über die Güte der durch diese Heuristiken gefundenen Rundreisen sagen kann. Zunächst ein negatives Resultat bezüglich der Möglichkeit, eine approximative Lösung für das symmetrische TSP zu finden.

(8.7) Satz. *Für ein beliebiges symmetrisches TSP bezeichnen wir mit T_{opt} eine optimale Tour. Gibt es ein $\varepsilon > 0$ und einen polynomialen Algorithmus H, der für jedes symmetrische TSP eine Tour T_H liefert mit*

$$c(T_{\mathrm{opt}}) \leq c(T_H) \leq (1+\varepsilon)c(T_{\mathrm{opt}}),$$

dann ist $P = \mathcal{NP}$, d. h. das ε-Approximationsproblem des symmetrischen TSP ist $\mathcal{NP}$-vollständig. △

Beweis. Wir wissen, dass das folgende Problem $\mathcal{NP}$-vollständig ist:

(HAM) Gibt es einen hamiltonschen Kreis in einem Graphen G?

Es ist klar, dass das ε-Approximationsproblem für das TSP in $\mathcal{NP}$ ist. Um zu zeigen, dass dieses Problem $\mathcal{NP}$-vollständig ist, beweisen wir folgendes: Wenn es einen Algorithmus H mit den obigen Eigenschaften gibt, dann gibt es auch einen polynomialen Algorithmus für (HAM).

Angenommen es existieren ein $\varepsilon > 0$ und ein polynomialer Algorithmus H mit obigen Eigenschaften. Sei $G = (V, E)$ ein beliebiger Graph mit $n = |V|$, und sei

$$M := \varepsilon n + 2.$$

Wir definieren ein TSP auf n Städten durch die folgenden Entfernungen:

$$c_{ij} = \begin{cases} 1 & \text{falls } ij \in E, \\ M & \text{sonst.} \end{cases}$$

Aufgrund dieser Definition gilt:

$$G \text{ hamiltonsch} \iff c(T_{\text{opt}}) = n.$$

Ist T eine Tour, die kein hamiltonscher Kreis in G ist, so gilt

$$c(T) \geq n - 1 + M = n - 1 + \varepsilon n + 2 > (1 + \varepsilon)n.$$

Sei nun T_H die von der Heuristik gefundene Lösung, und sei G hamiltonsch, so gilt $c(T_{\text{opt}}) = n \leq c(T_H) \leq (1 + \varepsilon)c(T_{\text{opt}}) = (1 + \varepsilon)n$. Da aber für jede Tour, die kein Hamiltonkreis in G ist, $c(T) > (1 + \varepsilon)n$ gilt, muss T_H ein hamiltonscher Kreis in G sein, d. h. wir können einen hamiltonschen Kreis in G in polynomialer Zeit finden. □

Die Situation des Satzes (8.7) trifft leider auf relativ viele kombinatorische Optimierungsprobleme zu. Das heißt, man kann in polynomialer Zeit nicht einmal eine feste Fehlerschranke garantieren. Ein Ausweg bleibt allerdings. Falls praktische Probleme vorliegen, sollte man versuchen, spezielle Strukturen zu finden und diese auszunutzen. Für die Theorie heißt das, man muss Spezialfälle des Problems bestimmen, für die Gütegarantien bewiesen werden können (und die möglichst die praxisrelevanten Beispiele umfassen).

Beim TSP ist der folgende Spezialfall von besonderer Praxisbedeutung. Wir nennen ein symmetrisches TSP *metrisch*, wenn für die Entfernungen die *Dreiecksungleichung* gilt, wenn also für alle Tripel i, j, k von Knoten gilt:

$$c_{ik} \leq c_{ij} + c_{jk}.$$

Bei Maschinensteuerungsproblemen (z.B. Bohren von Löchern in Leiterplatten) und bei geographischen Problemen ist die Dreiecksungleichung erfüllt. Das gilt jedoch häufig nicht für Entfernungstabellen in Straßenatlanten. Für euklidische TSP gilt (8.7) nicht.

(8.8) Satz. *Für metrische TSP und die Heuristiken aus (8.5) gelten die in Tabelle 8.2 angegebenen Gütegarantien. Die Laufzeitschranken gelten für beliebige Travelling Salesman Probleme.* △

Tabelle 8.2 ist wie folgt zu lesen. In der Spalte Laufzeit steht die Ordnung der Laufzeit, also ist z. B. NI ein $O(n^2)$-Algorithmus. In der Spalte ε besagt die dort angegebene Zahl, dass die zugehörige Heuristik immer eine Lösung liefert, deren Wert nicht größer als $(1 + \varepsilon)c(T_{\text{opt}})$ ist. Also weichen z. B. die Werte der von der Christofides-Heuristik gefundenen Lösungen höchstens um 50 % vom Optimalwert ab.

Name	ε	**Laufzeit**
Nächster Nachbar (NN)	$\frac{1}{2}(\lceil \log n \rceil - 1)$	n^2
Nearest Insert (NI)	1	n^2
Farthest Insert (FI)	$\geq \frac{1}{2}$	n^2
Cheapest Insert (CI)	1	$n^2 \log n$
Spanning-Tree (ST)	1	n^2
Christofides (CH)	$\frac{1}{2}$	n^3

Tabelle 8.2: Gütegarantien und Laufzeiten für TSP-Heuristiken

Beweis. Die Aussagen über die Laufzeiten folgen direkt aus den Darstellungen der Algorithmen in (8.5). Wir beweisen die Güteschranken von ST und CH.

Entfernen wir aus einer Tour eine Kante, so erhalten wir einen aufspannenden Baum. Da bei einem euklidischen TSP alle Kantengewichte nichtnegativ sind, folgt daraus, dass der Wert eines minimalen aufspannenden Baums B nicht größer ist als der der optimalen Tour T_{opt}.

Daher gilt für die Heuristik ST: $c(B_2) = \sum_{ij \in B} 2c_{ij} \leq 2c(T_{\text{opt}})$.

Die aus B_2 konstruierte Tour T entsteht dadurch, dass Wege in B_2 durch Kanten zwischen den Endknoten der Wege ersetzt werden. Da die Dreiecksungleichung gilt, ist der Wert dieser Kanten nicht größer als die Summe der Werte der Kanten der Wege. Daraus folgt $c(T) \leq c(B_2) \leq 2c(T_{\text{opt}})$, was zu zeigen war.

Bei der Christofides-Heuristik müssen wir den Wert des minimalen perfekten Matchings M abschätzen. Seien $i_1, \ldots, i_{2m}$ die ungeraden Knoten des aufspannenden Baumes B und zwar so numeriert, wie sie in T_{opt} vorkommen, d. h. $T_{\text{opt}} = (i_1, \alpha_1, i_2, \alpha_2, \ldots, \alpha_{2m-1}, i_{2m}, \alpha_{2m})$, wobei die α_i (möglicherweise leere) Folgen von Knoten sind. Wir betrachten nun die beiden Matchings $M_1 = \{i_1 i_2, i_3 i_4, \ldots, i_{2m-1} i_{2m}\}$, $M_2 = \{i_2 i_3, i_4 i_5, \ldots, i_{2m-2} i_{2m-1}, i_{2m} i_1\}$. Aufgrund der Dreiecksungleichung gilt $c(T_{\text{opt}}) \geq c(M_1) + c(M_2)$. Da M optimal ist, gilt somit $c(M) \leq \frac{1}{2} c(T_{\text{opt}})$. Wie vorher folgt daraus für die Christofides-Tour T: $c(T) \leq c(B_2) = c(B) + c(M) \leq c(T_{\text{opt}}) + \frac{1}{2} c(T_{\text{opt}}) = \frac{3}{2} c(T_{\text{opt}})$. □

Die Gütegarantie der Christofides-Heuristik ist die beste derzeit bekannte Gütegarantie für das symmetrische TSP mit Dreiecksungleichung. Bei weiterer Verschärfung von Bedingungen an die Zielfunktion kann man noch bessere Approximation erreichen.

Kann man aus der Gütegarantie ablesen, wie gut sich eine Heuristik in der Praxis verhält? Leider kann man diese Frage nicht uneingeschränkt bejahen. Es scheint so, dass zur Beurteilung immer noch praktische Tests an vielen und „repräsentativen“ (was auch immer das ist) Beispielen ausgeführt werden müssen. So wird z. B. in der Literatur relativ übereinstimmend berichtet, dass – bezüglich der hier vorgestellten Heuristiken – bis zu etwa 200 Knoten FI sehr häufig besser als die übrigen Verfahren ist. Für größere Probleme ist meistens die Christofides-Heuristik am besten, während FI die zweitbeste Heuristik ist. Bedenkt man allerdings, dass die Christofides-Heuristik als Unterprogramm ein (nicht trivial zu implementierendes) Verfahren zur Lösung von Matching Problemen benötigt, dann zeigt FI natürlich deutliche praktische Vorteile. Da der Christofides-Algorithmus

ja ohnehin nur eine Heuristik ist, kann man natürlich den Matching-Algorithmus durch eine Matching-Heuristik ersetzen. Dadurch verliert man zwar die globale Gütegarantie, erhält aber doch eine empirisch ordentliche Heuristik, die relativ leicht zu codieren ist (als Matching-Heuristik kann man z. B. den Greedy-Algorithmus wählen). Gleichfalls sollte bemerkt werden, dass die Spanning-Tree-Heuristik trotz der nicht schlechten Gütegarantie (empirisch) in der Praxis im Vergleich mit anderen Verfahren nicht so gut abschneidet. Das gleiche gilt für CI und NI. Eine sorgfältige empirische Analyse der hier genannten und anderer Heuristiken findet man in Reinelt (1994).

Damit wollen wir es bezüglich Eröffnungsverfahren bewenden lassen und bemerken, dass sich die hier dargestellten Ideen mit etwas Einfühlungsvermögen auf alle übrigen kombinatorischen Optimierungsprobleme übertragen lassen. Bei der Wahl des „lokalen Optimierungskriteriums“ ist es wichtig, nicht zu kurzsichtig zu wählen (NN ist z. B. eine relativ schlechte Heuristik). Man sollte hier globale Aspekte berücksichtigen. Dies ist z. B. bei FI gut gelöst, wie folgende (heuristische) Überlegung zeigt. Wählt man wie bei CI den lokalen bestmöglichen Einbau eines Knotens in den gegenwärtigen Kreis, so kann man in Fallen laufen, d. h. am Ende müssen einige ungünstig gelegene Knoten eingebaut werden, die in den bisherigen Kreis nicht „passen“. Bei FI versucht man, diese globalen Routenführungsfehler dadurch zu vermeiden, dass man immer denjenigen Knoten kostengünstigst einbaut, der gegenwärtig am weitesten entfernt liegt. Dadurch kann man am Ende nicht all zu viele Fehler machen.

8.3 Verbesserungsverfahren

Hat man durch ein Eröffnungsverfahren oder auf irgendeine andere Weise eine zulässige Lösung gefunden, so sollte man versuchen, die gegenwärtige Lösung durch Modifikationen zu verbessern. Die wichtigste Verfahrensklasse in der Kategorie der Verbesserungsheuristiken sind die *Austauschverfahren*. Die Idee hinter Austauschverfahren ist die folgende.

> Entferne einige Elemente aus der gegenwärtigen Lösung T, um eine Menge S zu erhalten. (Ist die Lösungsmenge ein Unabhängigkeitssystem, so ist S natürlich zulässig, i. A. (wie z. B. beim TSP) muss S keine zulässige Lösung sein.) Nun versuchen wir, alle möglichen zulässigen Lösungen, die S enthalten, zu erzeugen. Falls dies einen zu großen Rechenaufwand erfordert, generiert man entweder nach einem festen Schema eine Teilmenge aller zulässigen Lösungen, die S enthalten, oder man benutzt wieder ein lokales Optimalitätskriterium, um eine derartige Lösung zu bestimmen. Ist unter den erzeugten zulässigen Lösungen eine, die besser als T ist, nennen wir diese T und wiederholen die Prozedur. Gibt es keine bessere Lösung als T, entfernen wir andere Elemente aus T und verfahren analog. Wir beenden das Verfahren, wenn alle (nach einer vorgegebenen Regel) möglichen Austauschschritte keine bessere Lösung produziert haben.

Für das symmetrische TSP wollen wir drei Varianten dieses Austauschverfahrens genauer darstellen.

(8.9) Algorithmus (Austauschverfahren für das symmetrische TSP).

(a) **Zweier-Austausch (2-OPT)**.

1. Wähle eine beliebige Anfangstour $T = (i_1, i_2, \ldots, i_n)$.
2. Setze $Z := \{\{i_p i_{p+1}, i_q i_{q+1}\} \mid p+1 \neq q, p \neq q, q+1 \neq p, 1 \leq p, q \leq n\}$.
3. Für alle Kantenpaare $\{i_p i_{p+1}, i_q i_{q+1}\} \in Z$ führe aus:

   Ist $c_{i_p i_{p+1}} + c_{i_q i_{q+1}} > c_{i_p i_q} + c_{i_{p+1} i_{q+1}}$, setze $T := (T \setminus \{i_p i_{p+1}, i_q i_{q+1}\}) \cup \{i_p i_q, i_{p+1} i_{q+1}\}$ und gehe zu 2
4. Gib T aus.

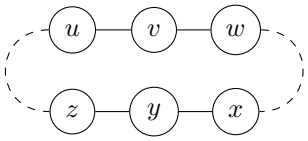
(b) **r-Austausch (r-OPT)**.

1. Wähle eine beliebige Anfangstour T.
2. Sei Z die Menge aller r-elementigen Teilmengen von T.
3. Für alle $R \in Z$ führe aus:

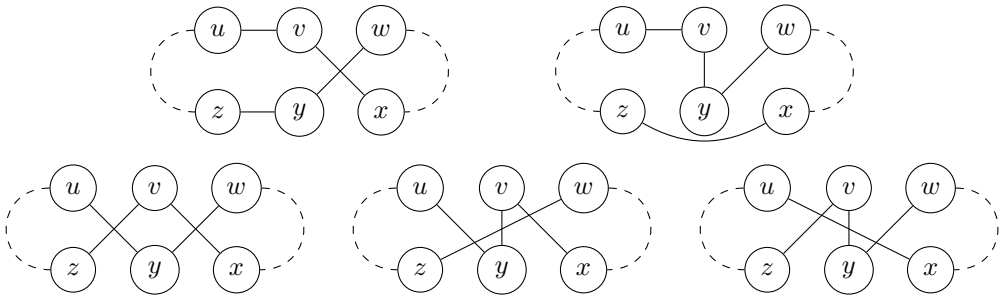
   Setze $S := T \setminus R$ und konstruiere alle Touren, die S enthalten. Gibt es unter diesen eine, die besser als T ist, nenne sie T und gehe zu 2.
4. Gib T aus.

(c) **Austausch zweier Knoten (2-NODE-OPT)**.

1. Wähle eine beliebige Anfangstour T.
2. Sei $Z = \{(v, y) \mid v \text{ und } y \text{ sind nicht benachbart auf } T\}$.
3. Für alle Knotenpaare $(v, y) \in Z$ führe aus: O. B. d. A. habe T die folgende Gestalt:



Konstruiere aus T die folgenden 5 Touren



Ist eine der Touren kürzer als T, nenne sie T und gehe zu 2.

4. Gib T aus. △

(8.10) Bemerkung.

(a) Für einen Durchlauf der Schleife 3 benötigen 2-OPT und 2-NODE-OPT $O(n^2)$ Schritte, während r-OPT $O(n^r)$ Schritte benötigt. Es ist nicht bekannt, ob die drei Heuristiken aus (8.9) eine polynomiale Laufzeit haben.

(b) Auch zur beweisbaren Gütegarantie lässt sich für die r-OPT-Verfahren nicht viel Positives sagen. Chandra et al. (1999) zeigen z. B. für metrische TSPs, dass 2-OPT nur eine Gütegarantie von $O(\sqrt{n})$ und dass r-OPT eine Gütegarantie von bestenfalls $O(n^{\frac{1}{2^k}})$ hat. △

Für $r \geq 4$ ist das r-Austauschverfahren aus Laufzeitgründen praktisch nicht durchführbar, aber auch bei $r = 3$ treten bei größeren TSP-Instanzen erhebliche Rechenzeitprobleme auf. Dagegen laufen 2-OPT und 2-NODE-OPT empirisch mit vernünftigen Rechenzeiten.

Es ist üblich, eine Tour, die durch einen r-Austausch nicht mehr verbessert werden kann, *r-optimal* zu nennen. r-optimale Touren sind sogenannte „lokale Optima“, aus denen man mit Hilfe des r-Austausches nicht „herauskommt“. (Achtung: Die hier übliche Verwendung des Begriffs „lokales Optimum“ ist etwas „salopp“. Derartige lokale Optima sind nicht lokal optimal in der üblichen Terminologie der Optimierungstheorie.)

Man beachte, dass 2-NODE-OPT als eine Mischung des 2-OPT-, 3-OPT- und 4-OPT-Verfahrens angesehen werden kann, wobei alle Zweiertausche aber nur wenige der Dreier- und Vierertausche ausgeführt werden. Die Zahl der Dreier- und Vierertausche wird so stark reduziert, dass die Laufzeit für Schritt 3 $O(n^2)$ bleibt, aber immerhin einige „vielversprechende“ Tausche vorgenommen werden. Daraus folgt, dass jede 2-NODE-optimale Tour auch 2-optimal ist.

In der Praxis sind die obigen Austauschverfahren sehr erfolgreich und können in der Regel die durch die Eröffnungsverfahren (8.5) gefundenen Lösungen verbessern. Erstaunlich dabei ist die (von vielen „Heuristikern„ gemachte) Beobachtung, dass die Austauschverfahren häufig bessere Lösungen liefern, wenn man sie mit zufälligen oder relativ schlechten Startlösungen beginnt, als wenn man mit guten heuristischen Lösungen (etwa durch FI gefundene) beginnt.

Startet man z. B. 2-OPT mit der durch NN gefundenen Lösung des Rheinlandproblems (8.6)(a_1), so wird der in Abbildung 8.4 gezeigte 2-Austausch vollzogen.

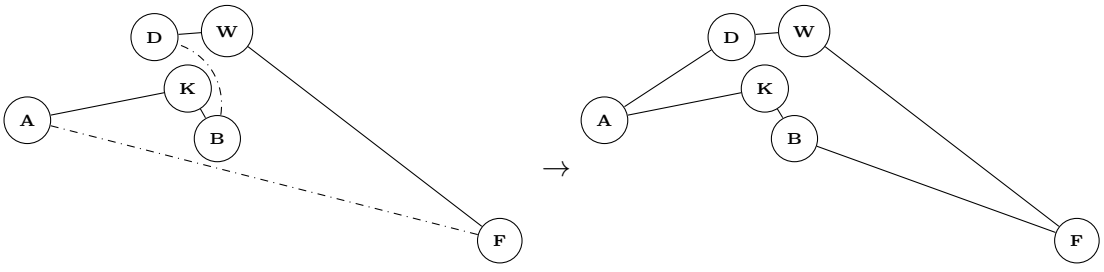


Abbildung 8.4: 2-OPT-Schritt am Rheinland-Problem

Diese Tour hat die Länge 617 km. Es wurde also eine Verbesserung um 81 km erzielt.

Die Austausch-Verfahren für das TSP und andere Kombinatorische Optimierungsprobleme sind stark von Lin und Kernighan (1973) propagiert worden. In ihrem Aufsatz zeigen sie z. B. dass eine dynamische Mischung von 2-OPT und 3-OPT mit $O(n^2)$ Laufzeit empirisch sehr gute Lösungen in akzeptabler Laufzeit liefert. Es ist vermutlich nicht falsch, wenn man die Lin-Kernigham-Heuristik (bzw. Varianten davon) als die beste derzeit bekannte TSP-Heuristik bezeichnet.

Es ist jedoch keineswegs trivial, sie so zu implementieren, dass sie effektiv und in akzeptabler Laufzeit arbeitet. Die derzeit (vermutlich) beste Variante dieses Verfahrens stammt von Keld Helsgaun. Informationen hierzu findet man u. a. im Stony Brook Algorithm Repository und auf Helsgauns Homepage. Mit seiner Heuristik wurde die beste bisher bekannte Lösung eines 1.904.711-Städte TSP (World TSP) gefunden. Animierte Bilder von der besten Lösung des World TSP findet man auf der Seite:

`http://www.math.uwaterloo.ca/tsp/world/pictures.html`

Eine gute Übersicht über Heuristiken für das Travelling Salesman Problem gibt der Aufsatz Johnson und Papadimitriou (1985). Im Aufsatz von Gilmore et al. (1985) wird gezeigt, dass gewisse Heuristiken für Spezialfälle des TSP immer Optimallösungen liefern.

Ein Verbesserungsverfahren, das einige Aufmerksamkeit erregt hat, ist die Methode des „Simulated Annealing“. Dies ist ein Verfahren, das erwachsen ist aus Simulationsmethoden der statistischen Mechanik. In dieser physikalischen Theorie untersucht man u. a. Phasenübergänge wie Kristallisation von Flüssigkeiten oder das Entstehen von Magneten. Derartige Phänomene treten bei gewissen kritischen Temperaturen auf (z. B. friert Wasser bei 0° C), und man möchte wissen, wie sich die Materie in diesem kritischen Temperaturbereich organisiert, um z. B. Kristalle zu bilden. Da man für verschiedene technische Prozesse Kristalle ohne Strukturdefekte benötigt, möchte man wissen, wie man reale Flüssigkeiten kühlt und wärmt, so dass möglichst reine Kristalle entstehen.

Bevor man heutzutage reale Experimente mit derartigen physikalischen Systemen ausführt, werden meistens – besonders dann, wenn die Experimente teuer sind – Computersimulationen dieser Experimente durchgeführt. Hierzu wird ein abstraktes Modell des physikalischen Systems entworfen und implementiert und (zufälligen) Änderungen unterworfen, die z. B. in der Realität Kühlvorgängen entsprechen. Bei diesen Computerexperimenten lernt man, die realen Vorgänge besser zu verstehen, man kann möglicherweise bereits eine Theorie aufstellen, und man braucht sicherlich durch die umfangreichen Voruntersuchungen sehr viel weniger reale Experimente um z. B. die Herstellung von reinen Kristallen in den Griff zu bekommen.

Es ist – unter einigen Zusatzannahmen – möglich, viele kombinatorische Optimierungsprobleme als Realisierung physikalischer Systeme und die Optimierungsvorschrift als Energieminimierung anzusehen. Daher sind einige Physiker, insbesondere angeregt durch den Aufsatz Kirkpatrick et al. (1983) auf die Idee gekommen, kombinatorische Optimierungsprobleme mit den relativ ausgereiften Simulationstechniken der statischen Mechanik zu behandeln. Dies ist der Grund dafür, dass man bei den Darstellungen dieser Methoden viel physikalisches Vokabular findet, hinter dem man i. A. mehr vermutet, als einige recht simple heuristische Ideen.

Sorgfältige Implementierungen zeigen, dass Simulated Annealing bei einigen interessanten Problemen relativ konsistent sehr gute heuristische Lösungen erzeugt. Übereinstimmende Erfahrung ist jedoch, dass Simulated Annealing viel Parameter- und Feintuning und bis zur Generierung guter Lösungen sehr hohen Zeitaufwand erfordert. Allein aus dem letzten Grunde ist Simulated Annealing für viele (speziell große) praktische Probleme nicht sonderlich geeignet.

Nach der langen Vorrede nun eine kurze Beschreibung des Verfahrens – ohne physikalisches Brimborium.

(8.11) Algorithmus (Simulated-Annealing-Verfahren (Überblick)). Wir nehmen an, dass ein kombinatorisches Minimierungsproblem $(E, \mathcal{I}, c)$ vorliegt.

(1) Wähle mehrere Startheuristiken zur approximativen Lösung von $(E, \mathcal{I}, c)$. Seien $I_1, \ldots, I_k$ die hierbei gefundenen Lösungen. Sei I_0 die beste dieser Lösungen.

(2) Wähle ein (oder mehrere) Verbesserungsverfahren.

(3) Initialisiere zwei Listen L und L' von jeweils $k \geq 1$ Lösungen von $(E, \mathcal{I}, c)$. Setze $I_1, \ldots, I_k$ auf Liste L.

(4) Für alle Lösungen I auf Liste L führe aus:

   (a) Wende eines oder alle Verbesserungsverfahren auf I an (oder wähle eines der Verfahren zufällig oder wähle eine aus I zufällig erzeugte neue Lösung).

   (b) Wird eine Lösung I' gefunden, deren Wert besser als der von I ist, setze I' auf L'. Gilt $c(I') < c(I_0)$, setze $I_0 := I'$.

   (c) Wird keine Lösung produziert, die besser als I ist, setze die beste produzierte Lösung I' mit Wahrscheinlichkeit p auf L'. (Wichtig: p sollte während des Verfahrens variieren, und die Wahrscheinlichkeit des Akzeptierens schlechter Lösungen sollte mit zunehmender Ausführungsdauer abnehmen!)

(5) Ist L' leer, STOP und gib I_0 aus, andernfalls setze $L := L'$, $L' := \emptyset$ und gehe zu (4). △

Das oben beschriebene Verfahren hat sehr viele offene Parameter, von deren Wahl der Erfolg stark abhängt. Leider ist es nicht einfach, herauszufinden, wann welche Strategie bei welchen Problemen zu einer guten Heuristik führt.

Die Idee hinter dem Verfahren ist einfach erklärt. Man bearbeitet parallel mehrere Lösungen, um auf verschiedenen Wegen Fortschritte zu erzielen und so u. U. in mehrere „lokale Minima“ hineinzulaufen, von denen vielleicht eines das globale Minimum ist. (Manche Verfahren des Simulated Annealing arbeiten jedoch nur mit $k = 1$, also einer gegenwärtigen Lösung.) Der meiner Meinung nach wichtigste Aspekt besteht darin, dass man gewillt ist, gelegentlich auch Verschlechterungen hinzunehmen. Hinter diesem Schritt steckt die folgende Erfahrung. Verbesserungsverfahren laufen relativ schnell in Fallen (bzw. lokale Minima), aus denen die Verbesserungsvorschriften nicht herausführen. Geht man jedoch auf eine schlechtere Lösung zurück, so kann man durch einen

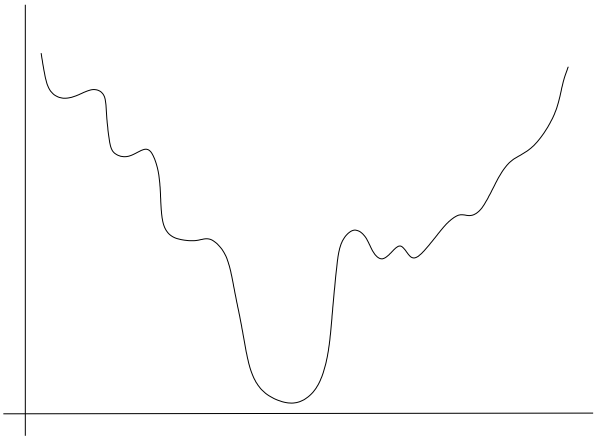


Abbildung 8.5: Suggestive Darstellung zu Simulated Annealing

erneuten und weitere Austausche u. U. zu einer wesentlich besseren Lösung gelangen. Hierzu liefert Abbildung 8.5 eine suggestive Anschauung.

Trägt man auf der x-Achse die Lösungen auf und auf der y-Achse deren Wert und betrachtet man die Methode, in der linken oberen Ecke einen Ball laufen zu lassen, als Verbesserungsheuristik, so ist klar, dass der Ball in Abbildung 8.5 häufig in den Nebentälern (lokale Minima) hängen bleibt. Stößt man den Ball jedoch zwischendurch gelegentlich an, so kann er – abhängig von der Stoßrichtung und -stärke – aus dem Tal herauskatapultiert werden und in ein tieferes Tal rollen. Dies „zeigt“, dass gelegentliches Verschlechtern langfristig zu besseren Lösungen führen kann.

All diese Überlegungen sind jedoch nur überzeugende Prinzipien, bei denen nicht direkt klar ist, ob sie sich auch in effiziente heuristische Verfahren übersetzen lassen. Methoden des Typs (8.11) sind sehr nützlich in Bezug auf kombinatorische Optimierungsprobleme mit wenig untersuchten Strukturen und Nebenbedingungen. Nach meinen Erfahrungen sind Verfahren dieser Art (insbesondere wegen der hohen Laufzeiten) in der Regel den recht sophistifizierten Heuristiken für intensiv studierte Probleme wie das TSP unterlegen. Inzwischen gibt es eine Vielzahl von Aufsätzen zu *Simulated Annealing* und dessen Varianten. Man schaue z. B. nur das Journal of Heuristics durch.

Zwei gute „Computational Studies“, die über Erfahrungen mit Simulated Annealing berichten, sind Johnson et al. (1989) und Johnson et al. (1991). In diesen Papern werden einige konkrete Implementierungen des allgemeinen Verfahrens (8.11) angegeben, bezüglich verschiedener kombinatorischer Optimierungsprobleme (wie das weiter oben vorgestellte Leiterplatten-Platzierungsproblem, das Subset-Sum-Problem, das Knotenfärbungsproblem für Graphen und das TSP) getestet und mit bekannten Heuristiken für diese Probleme verglichen. Soweit mir bekannt ist, sind die Erfahrungen „durchwachsen“.

Ein Buch, das Simulated Annealing im Detail beschreibt und insbesondere den stochastischen Hintergrund erklärt, ist van Laarhoven und Aarts (1987).

Weitere Verfahren, die häufig „Meta-Heuristiken“ genannt werden, wurden in der Vorlesung skizziert:

- Tabu-Search
- Genetische Algorithmen
- Evolutionsmethoden
- Neuronale Netze
- Ameisensysteme
- Space-Filling Curves.

Literaturhinweise hierzu findet man u. a. auf der bereits zitierten TSPBIB-Homepage:

`http://frcatel.fri.utc.sk/users/pesko/Studenti/TSPBIB.htm`

Ein gutes Buch mit Übersichtsartikeln zu den verschiedenen heuristischen Techniken ist Aarts und Lenstra (1997), in dem z. B. Johnson und McGeoch eine weitere Studie zu TSP-Heuristiken (A Case Study in Local Optimization) vorlegen, die viele der hier angedeuteten Heuristiken näher erklärt, auf Implementierungsaspekte eingeht und die Rechenstudien mit Verfahrensvergleichen enthält.

8.4 Gütemaße für Heuristiken

In diesem Abschnitt wollen wir die Gütekriterien zur Bestimmung des Approximationsverhaltens von Heuristiken untersuchen.

Wir haben bereits Beispiele für heuristische Verfahren angegeben, für die gewisse Gütegarantien bewiesen werden können. Wir haben auch gesehen, dass es bei manchen Problemen überhaupt nicht möglich ist, mit polynomialem Rechenaufwand zu garantieren, dass eine fast optimale Lösung gefunden werden kann (falls $\mathcal{P} \neq \mathcal{NP}$). Ferner gibt es Probleme, bei denen bestmögliche Worst-Case Schranken existieren, die mit polynomialen Verfahren erreicht aber nicht verbessert werden können. Um diese (und andere) Fälle besser klassifizieren zu können, hat sich die folgende Terminologie gebildet.

(8.12) Definition. *Sei Π ein kombinatorisches Optimierungsproblem, und es sei $\Pi' := \{P \in \Pi \mid P$ hat eine zulässige Lösung$\}$. Für jedes Problembeispiel $P \in \Pi'$ bezeichne $c_{\text{opt}}(P)$ den Wert einer Optimallösung von P. Ferner sei A ein Algorithmus, der für jedes Problembeispiel $P \in \Pi'$ eine zulässige Lösung liefert. Den Wert dieser Lösung bezeichnen wir mit $c_A(P)$. Um aufwendige Fallunterscheidungen zu umgehen, nehmen wir im Weiteren $c_{\text{opt}}(P) > 0$ für alle $P \in \Pi'$ an und setzen $c_{\text{opt}}(P) := c_A(P) := 1$ für alle Problembeispiele $P \in \Pi \setminus \Pi'$.*

(a) *Sei $\varepsilon > 0$ eine fest vorgegebene Zahl. Falls Π ein Maximierungsproblem ist, gelte zusätzlich $\varepsilon \leq 1$. Gilt für jedes Problembeispiel $P \in \Pi$*

$$R_A(P) := \frac{|c_A(P) - c_{\text{opt}}(P)|}{c_{\text{opt}}(P)} \leq \varepsilon,$$

so heißt A ε-approximativer Algorithmus*, und die Zahl ε heißt* Gütegarantie *von A.*

(b) *Sei A ein ε-approximativer Algorithmus für Π.*

(b_1) *Ist Π ein Maximierungsproblem, dann heißt $1-\varepsilon$ die* Worst-Case-Schranke *von A (denn dann gilt $c_A(P) \geq (1-\varepsilon)c_{\text{opt}}(P)$).*

(b_2) *Ist Π ein Minimierungsproblem, dann heißt $1+\varepsilon$ die* Worst-Case-Schranke *von A (denn dann gilt $c_A(P) \leq (1+\varepsilon)c_{\text{opt}}(P)$).*

(c) *Ein* Approximationsschema (AS) *für Π ist ein Algorithmus A, der zwei Inputs hat, nämlich ein Problembeispiel $P \in \Pi$ und eine rationale Zahl $\varepsilon > 0$, und der für jedes $P \in \Pi$ und jedes $\varepsilon > 0$ eine Lösung produziert, die $R_A(P) \leq \varepsilon$ erfüllt.*

(d) *Ein Approximationsschema A für Π heißt* polynomiales Approximationsschema (PAS)*, wenn die Laufzeit von A polynomial ist in der Inputlänge von $P \in \Pi$.*

(e) *Ein Approximationsschema A für Π heißt* voll-polynomiales Approximationsschema (FPAS)*, wenn die Laufzeit von A polynomial in $\langle P \rangle + \frac{1}{\varepsilon}$ ist, also polynomial in der Inputlänge von $P \in \Pi$ und polynomial in $\frac{1}{\varepsilon}$ ist.* △

Die Unterscheidung zwischen „Gütegarantie“ und „Worst-Case-Schranke“ ist etwas künstlich und häufig werden – auch von mir – die beiden Begriffe durcheinander gebracht.

(8.13) Beispiel.

(a) Ist q der Rangkoeffizient eines Unabhängigkeitssystems $(E, \mathcal{I}, c)$, so liefert der Greedy-Algorithmus, siehe (2.14), die Gütegarantie $1-q$ und ist somit ein Algorithmus mit Worst-Case-Schranke q für Maximierungsprobleme über allgemeine Unabhängigkeitssysteme.

(b) Die Christofides-Heuristik (8.5)(f) ist ein $\frac{1}{2}$-approximatives Verfahren für das euklidische symmetrische TSP.

(c) Ist $P \in \Pi$, hat P die Inputlänge n und hat ein Algorithmus A z. B. die Laufzeit $O(n^{1/\varepsilon})$, so ist A ein polynomiales Approximationsschema. Hat A z. B. die Laufzeit $O\left(n^k \cdot \left(\frac{1}{\varepsilon}\right)^l\right)$ (mit Konstanten k und l), so ist A ein voll-polynomiales Approximationsschema. △

Keine der Heuristiken, die wir bisher kennengelernt haben, ist ein polynomiales oder gar voll-polynomiales Approximationsschema. Die Gütegarantien gelten nur für ein bestimmtes ε (und alle größeren Werte).

Es stellt sich natürlich sofort die Frage, ob es für alle schwierigen Optimierungsprobleme möglich ist, ein polynomiales bzw. voll-polynomiales AS zu entwickeln. Wir werden dies im folgenden theoretisch untersuchen.

Man beachte, dass ein FPAS nicht polynomial im Gesamtinput ist. Nach Definition ist es zwar polynomial in der Inputlänge von $P \in \Pi$ und in $\frac{1}{\varepsilon}$, jedoch benötigt man zur Darstellung der rationalen Zahl $\varepsilon = \frac{p}{q}$ (bzw. $\frac{1}{\varepsilon}$) nur $\langle \varepsilon \rangle \approx \log p + \log q$ und nicht $p+q$ Speicherplätze.

(8.14) Satz. *Sei* Π *ein kombinatorisches Optimierungsproblem. Gibt es für* Π *ein FPAS, das auch polynomial in* $\langle\varepsilon\rangle$ *ist, so gibt es einen polynomialen Algorithmus zur Lösung aller Problembeispiele in* Π. △

Beweis. Gibt es überhaupt ein FPAS, sagen wir A, so sind die Kodierungslängen sowohl des Wertes der Lösung $c_A(P)$ von A als auch des Optimalwertes $c_{\text{opt}}(P)$ polynomial in der Inputlänge von P. Ist A polynomial in $\langle\varepsilon\rangle$, so können wir einen polynomialen Algorithmus B für Π wie folgt konstruieren.

1. Setze $\varepsilon := \frac{1}{2}$ und wende A auf $P \in \Pi$ an. Wir erhalten einen Wert $c_{A_\varepsilon}(P)$.

2. Setze

$$\text{im Maximierungsfall} \quad \delta := \frac{1}{2c_{A_\varepsilon}(P)+1},$$
$$\text{im Minimierungsfall} \quad \delta := \frac{1}{c_{A_\varepsilon}(P)+1}.$$

3. Wende A auf P mit Genauigkeitsparameter δ an. Wir erhalten eine Lösung $c_{A_\delta}(P)$.

Sei B der durch 1, 2 und 3 beschriebene Algorithmus, so ist die Laufzeit von B offensichtlich polynomial in der Inputlänge von P, denn $\varepsilon = \frac{1}{2}$ ist eine Konstante, somit sind $c_{A_\varepsilon}(P)$ und δ Größen deren Kodierungslängen polynomial in der Inputlänge von P sind. Folglich ist auch Schritt 3 polynomial in der Inputlänge von P.

Wir können o. B. d. A. annehmen, dass die Zielfunktion ganzzahlig ist. Somit sind auch die Werte der optimalen Lösungen und der durch B erzeugten Lösung ganzzahlig. Wir zeigen nun, dass $c_{\text{opt}}(P) = c_{A_\delta}(P)$ gilt.

Ist Π ein Maximierungsproblem, so gilt:

$$\begin{aligned} c_{A_\varepsilon}(P) \geq \frac{1}{2}c_{\text{opt}}(P) &\implies 2c_{A_\varepsilon}(P)+1 > c_{\text{opt}}(P) \\ c_{A_\delta}(P) \geq (1-\delta)c_{\text{opt}}(P) &= \Big(1-\frac{1}{2c_{A_\varepsilon}(P)+1}\Big)c_{\text{opt}}(P) \\ &> \Big(1-\frac{1}{c_{\text{opt}}(P)}\Big)c_{\text{opt}}(P) \\ &= c_{\text{opt}}-1. \end{aligned}$$

Da $c_{A_\delta}(P)$ ganzzahlig und echt größer als $c_{\text{opt}}-1$ ist, folgt $c_{A_\delta}(P) = c_{\text{opt}}(P)$.

Ist Π ein Minimierungsproblem, so erhalten wir:

$$\begin{aligned} c_{\text{opt}}(P) < c_{A_\varepsilon}(P)+1 &\implies \frac{1}{c_{\text{opt}}(P)} > \frac{1}{c_{A_\varepsilon}(P)+1} \\ c_{\text{opt}}(P) \leq c_{A_\delta}(P) \leq (1+\delta)c_{\text{opt}}(P) &= \Big(1+\frac{1}{c_{A_\varepsilon}(P)+1}\Big)c_{\text{opt}}(P) \\ &< \Big(1+\frac{1}{c_{\text{opt}}(P)}\Big)c_{\text{opt}}(P) \\ &= c_{\text{opt}}(P)+1. \end{aligned}$$

Aufgrund der Ganzzahligkeit folgt daraus die Behauptung. □

In einem gewissen Sinne ist also ein FPAS das beste, was man als appoximatives Verfahren für ein $\mathcal{NP}$-vollständiges Problem erwarten kann.

Wir führen nun weitere Maßzahlen ein, die im Zusammenhang mit der Analyse approximativer Algorithmen interessant sind. (Die Definition dieser Zahlen ist in der Literatur nicht einheitlich.) Wir hatten in (8.12)(a) die Größe

$$R_A(P) := \frac{|c_A(P) - c_{\text{opt}}(P)|}{c_{\text{opt}}(P)}$$

für ein Problem $P \in \Pi$ definiert. Wir setzen nun:

(8.15) Definition. *Sei* Π *ein Optimierungsproblem.*

(a) *Ist* A *ein approximativer Algorithmus für* Π*, dann heißt*

$$R_A := \inf\{\varepsilon > 0 \mid R_A(P) \leq \varepsilon \text{ für alle } P \in \Pi\}$$

die absolute Gütegarantie von A*, und*

$$R_A^\infty := \inf\{\varepsilon > 0 \mid \text{Es gibt ein } n_0 \in \mathbb{N} \text{ so dass } R_A(P) \leq \varepsilon \text{ für alle } P \in \Pi \text{ mit } c_{\text{opt}} \geq n_0\}$$

heißt die asymptotische Gütegarantie von A*. (Bei Maximierungsproblemen* Π *geschieht die Infimumsbildung über alle* ε *mit* $0 < \varepsilon < 1$*.)*

(b) *Der Wert*

$$R_{\min}(\Pi) := \inf\{\varepsilon > 0 \mid \text{Es gibt einen polynomialen Approximationsalgorithmus } A \text{ mit } R_A^\infty = \varepsilon\}$$

(bzw. Infimum über $0 < \varepsilon < 1$ *bei Maximierungsproblemen) heißt die* bestmögliche asymptotische Gütegarantie von Π*. Es ist* $R_{\min}(\Pi) = \infty$*, falls kein solches* ε *existiert.* $\triangle$

$R_{\min}(\Pi)$ ist natürlich eine der wichtigsten Größen dieser Analysetechniken, da durch $R_{\min}(\Pi)$ die bestmögliche Approximierbarkeit beschrieben wird.

(8.16) Bemerkung. Für $R_{\min}(\Pi)$ sind drei Fälle von Bedeutung.

(a) $R_{\min}(\Pi) = \infty$,
d. h. für Maximierungsprobleme, dass für kein $0 < \varepsilon < 1$ ein polynomialer ε-approximativer Algorithmus existiert, und für Minimierungsprobleme, dass für kein $\varepsilon > 0$ ein polynomialer ε-approximativer Algorithmus existiert.

(b) $R_{\min}(\Pi) > 0$,
d. h. es gibt eine Schranke für die Gütegarantie, die mit polynomialem Aufwand nicht überwunden werden kann.

(c) $R_{\min}(\Pi) = 0$, d. h. das Optimum kann durch polynomiale Algorithmen beliebig approximiert werden. Offensichtlich kann jedoch nur ein Problem mit $R_{\min}(\Pi) = 0$ ein PAS oder ein FPAS haben. Zum Entwurf von PAS bzw. FPAS für ein Problem Π müssen wir also zunächst die asymptotische Gütegarantie kennen. Hier gibt es vier Unterfälle:

(c_1) Es gibt ein PAS für Π.

(c_2) Es gibt ein FPAS für Π.

(c_3) Es gibt einen polynomialen Approximationsalgorithmus A mit $R_A^\infty = 0$.

(c_4) Es gibt einen polynomialen Approximationsalgorithmus A mit $|c_A(P) - c_{\text{opt}}(P)| \leq K$ für ein festes K und alle $P \in \Pi$.

Eine solche Garantie wie in (c_4) nennt man auch *Differenzgarantie.* △

Offensichtlich ist die beste aller Gütegarantien, die man erwarten darf, durch (c_4) beschrieben.

Der Fall (8.16)(a) trifft auf das symmetrische Travelling-Salesman-Problem zu, wie Satz (8.7) zeigt. Wir wollen nun ein Problem einführen, eine bestmögliche Heuristik angeben und zeigen, dass für dieses Problem (8.16)(b) zutrifft.

(8.17) Beispiel (Das *k*-Zentrumsproblem). Gegeben sei ein vollständiger Graph $K_n = (V, E)$ mit Kantengewichten c_e für alle $e \in E$. Gesucht ist eine Knotenmenge (die Zentren) $S \subseteq V$ mit höchstens k Elementen, so dass

$$c(S) := \max_{i \in V} \min_{j \in S} c_{ij}$$

minimal ist. △

Falls $k = |V|$ gilt, setzt man natürlich $S = V$. Anwendungen dieses Standortproblems sind offensichtlich. Man möchte z. B. höchstens k Feuerwehrdepots so verteilen, dass die maximale Entfernung irgendeines möglichen Brandherdes zu einem Depot möglichst klein wird. Damit soll z. B. gewährleistet werden, dass jeder Brandherd in einer bestimmten maximalen Zeit erreichbar ist.

Dieses Problem ist $\mathcal{NP}$-schwer, selbst dann, wenn die Dreiecksungleichung $c_{ij} + c_{jk} \geq c_{ik}$ für alle $i, j, k \in V$ gilt. Man kann sogar zeigen:

(8.18) Satz. *Es gibt ein ε mit $0 < \varepsilon < 1$ und einen polynomialen Algorithmus A, der für jedes k-Zentrumsproblem mit Dreiecksungleichung eine Lösung S_A liefert, die bezüglich der Optimallösung S_{opt} folgende Gütegarantie hat*

$$c(S_{\text{opt}}) \leq c(S_A) \leq (1 + \varepsilon) c(S_{\text{opt}}),$$

genau dann, wenn $\mathcal{P} = \mathcal{NP}$ gilt. △

Beweis. Siehe Hsu und Nemhauser (1979). □

Mit anderen Worten: Das ε-Approximationsproblem für das k-Zentrumsproblem mit Dreiecksungleichung ist für $0 < \varepsilon < 1$ $\mathcal{NP}$-vollständig. Bemerkenswert ist nun, dass man die Gütegarantie $\varepsilon = 1$ in polynomialer Zeit tatsächlich erreichen kann. Wir betrachten dazu den folgenden Algorithmus.

(8.19) Algorithmus (1-approximative Heuristik für das k-Zentrumsproblem).

Eingabe: Vollständiger Graph $K_n = (V, E)$, mit Gewichten c_e für alle $e \in E$.
Ausgabe: Eine Menge $S \subseteq V$ mit $|S| \leq k$.

1. Falls $k = |V|$, dann gib V aus und STOP.

2. Wir ordnen die m Kanten so, dass $c_{e_1} \leq c_{e_2} \leq \cdots \leq c_{e_m}$ gilt.

3. Für jeden Knoten $v \in V$ legen wir eine Adjazenzliste ADJ(v) an, auf der die Nachbarn u von v in nicht-absteigender Reihenfolge bezüglich der Kantengewichte c_{uv} auftreten. Mit $G_i = (V, E_i)$ bezeichnen wir den Untergraphen von K_n mit Kantenmenge $E_i = \{e_1, e_2, \ldots, e_i\}$.

4. LOW := 1 (bedeutet: S kann die Menge V sein).
   HIGH := m (bedeutet: S besteht nur aus einem Element).

5. (Binäre Suche)
   Ist HIGH = LOW +1, gehe zu 9.

6. Setze $i := \lfloor \frac{\text{HIGH} + \text{LOW}}{2} \rfloor$ und konstruiere $G_i = (V, E_i)$ und die zugehörigen (sortierten) Adjazenzlisten ADJ$_i(v)$. Setze $S := \emptyset$ und $T := V$.

7. Ist $T \neq 0$, so wähle einen Knoten $v \in T$ und führe aus:

   $$S := S \cup \{v\},$$
   $$T := T \setminus (\{v\} \cup \{w \mid w \text{ Nachbar von } v \text{ in } G_i\})$$

   und gehe zu 7. (Achtung: Schritt 7 ist nichts anderes als der Greedy-Algorithmus zur Bestimmung einer stabilen Menge in G_i.)

8. Falls $|S| \leq k$, dann setze HIGH := i und $S' := S$, andernfalls LOW := i.
   Gehe zu 5.

9. Gib S' aus. △

Hochbaum und Shmoys (1985) haben gezeigt:

(8.20) Satz. *Der Algorithmus (8.19) liefert für k-Zentrumsprobleme, bei denen die Dreiecksungleichung gilt, eine Knotenmenge $S \subseteq V$, $|S| \leq k$, mit $c(S) \leq 2c(S_{\text{opt}})$ in $O(|E| \log |E|)$ Zeit.* △

Dieser Algorithmus – eine Mischung aus dem Greedy-Algorithmus und binärer Suche – ist also in der Tat bestmöglich unter allen approximativen Algorithmen für das k-Zentrumsproblem mit Dreiecksungleichung. Aus (8.20) und (8.18) folgt:

(8.21) Korollar. *Sei* Π *das k-Zentrumsproblem mit Dreiecksungleichung, dann gilt*

$$R_{\min}(\Pi) = 1. \qquad \triangle$$

8.5 Weitere Heuristiken

In diesem Abschnitt wollen wir (relativ unsystematisch) einige weitere Beispiele für Heuristiken angeben und sie analysieren. Hierbei soll die Technik der Analyse von Heuristiken geübt werden, und gleichzeitig werden auch einige weitere kombinatorische Optimierungsprobleme vorgestellt.

Heuristiken liefern bei Maximierungsproblemen (bzw. Minimierungsproblemen) untere (bzw. obere) Schranken für den Wert einer Optimallösung. Um die Güte einer heuristischen Lösung abschätzen zu können, bedient man sich häufig weiterer Heuristiken, die dann obere (bzw. untere) Schranken für den Optimalwert liefern. Diese Heuristiken nennt man *duale Heuristiken* und spricht dann auch von *primalen Heuristiken*, um die dualen Heuristiken von den oben eingeführten Verfahren zu unterscheiden. Das Wort „primal“ in diesem Zusammenhang kommt daher, dass diese Verfahren zulässige Lösungen für das Ausgangsproblem, also in der LP-Theorie das primale Problem liefern. Die Lösungen sind also primal zulässig.

Maschinenbelegung mit unabhängigen Aufgaben

Wir betrachten das folgende Problem der Maschinenbelegungsplanung:

(8.22) Bemerkung (Parallel-Shop-Problem).

(a) Gegeben seien m identische *Maschinen* oder Prozessoren

$$M_1, M_2, \ldots, M_m$$

(z. B. Drucker eines Computers, Offset-Drucker, Walzstraßen, Pressen, Stanzen).

(b) Gegeben seien n *Aufgaben* (oder Aufträge oder Operationen oder Jobs)

$$T_1, T_2, \ldots, T_n$$

(z. B. Druckjobs, -aufträge, zu walzende Profile, Press- und Stanzaufträge).

(c) Jeder Auftrag hat eine *Ausführungszeit* (oder Bearbeitungszeit)

$$t_1, t_2, \ldots, t_n, \quad \text{(in Zeiteinheiten)},$$

und jeder Auftrag benötigt nur eine Maschine zu seiner Bearbeitung.

(d) Die Maschinen arbeiten parallel, und jede kann nur eine Aufgabe gleichzeitig erledigen. Hat eine Maschine die Ausführung einer Aufgabe begonnen, so führt sie diese bis zum Ende der Bearbeitungszeit durch. △

Ein Maschinenbelegungsproblem dieser Art kann man einfach durch die Angabe einer Zahlenfolge auf folgende Weise beschreiben:

$$m, n,\ t_1, t_2, \ldots, t_n.$$

(8.23) Beispiel. Wir betrachten die Zahlenfolge

$$3,\ 11,\quad 2,\ 4,\ 3,\ 4,\ 4,\ 5,\ 2,\ 1,\ 4,\ 3,\ 2.$$

Das heißt, wir haben $m = 3$ Maschinen und $n = 11$ Aufträge, $T_1, \ldots, T_{11}$ zu bearbeiten. Ein möglicher *Belegungsplan* der drei Maschinen wäre der folgende:

$$\begin{aligned} &M_1 \text{ bearbeitet } T_1, T_2, T_3, T_4,\\ &M_2 \text{ bearbeitet } T_5, T_6, T_9,\\ &M_3 \text{ bearbeitet } T_7, T_8, T_{10}, T_{11}. \end{aligned}$$

Einen solchen Belegungsplan kann man bequem in einem Balkendiagramm wie folgt darstellen.

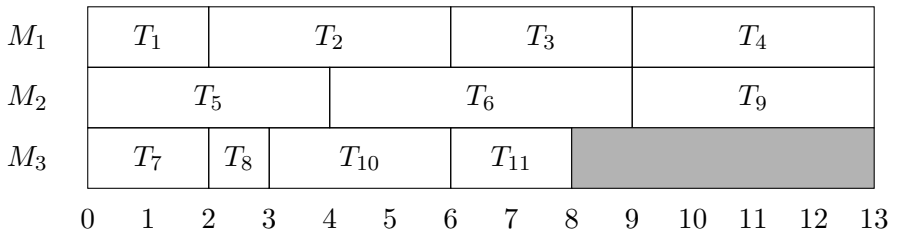


Abbildung 8.6: Belegungsplan in Beispiel (8.23)

Insgesamt benötigt man also bei diesem Belegungsplan zur Bearbeitung aller Aufträge 13 Zeiteinheiten, wobei jedoch die dritte Maschine 5 Zeiteinheiten leer steht. △

Ein sinnvolles und für die Praxis relevantes Optimierungsproblem besteht nun darin, einen Belegungsplan zu finden, so dass der Gesamtauftrag so früh wie möglich fertiggestellt ist. Zur vollständigen Formulierung von (8.22) fahren wir also wie folgt fort:

(8.22) Bemerkung (Fortsetzung).

(e) Ist ein Belegungsplan gegeben, so bezeichnen wir mit

$$C_1, C_2, \ldots, C_m$$

die *Fertigstellungsdauer (-zeit)* der auf den Maschinen $M_1, \ldots, M_m$ ausgeführten Aufträge.

(f) **Optimierungsziel:** Finde einen Belegungsplan, so dass die größte Fertigstellungsdauer so klein wie möglich ist. Dafür schreiben wir:

$$\min_{\text{Belegungspl.}} \max_{i=1,\ldots,m} C_i$$

oder kurz: $\min C_{\max}$, mit $C_{\max} = \max\{C_i \mid i = 1, \ldots, m\}$. △

Der Belegungsplan in Abb. 8.7 für das Beispiel (8.23) ergibt eine bessere Lösung als der in Abb. 8.6 angegebene.

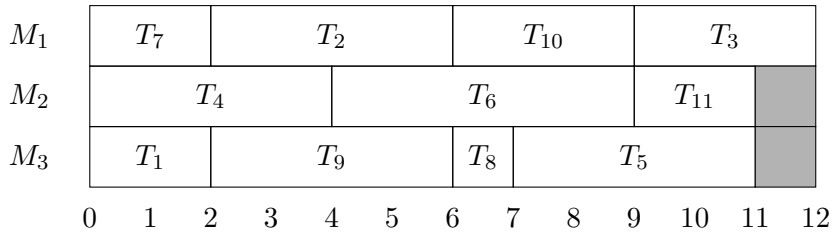


Abbildung 8.7: Besserer Belegungsplan für Beispiel (8.23)

In Abb. 8.7 werden insgesamt nur 12 Zeiteinheiten benötigt. Dieser Belegungsplan ist sogar optimal, denn insgesamt sind 34 Zeiteinheiten auf 3 Maschinen zu verteilen, und es ist offenbar unmöglich, 34 Zeiteinheiten auf 3 Maschinen zu jeweils 11 Zeiteinheiten Bearbeitungsdauer aufzuteilen. Das hier vorgestellte Maschinenbelegungsproblem (8.22) ist schwierig im Sinne der Komplexitätstheorie. Selbst das Problem mit 2 parallelen identischen Maschinen ist bereits $\mathcal{NP}$-vollständig.

Es ist daher sinnvoll, nach heuristischen Verfahren zu suchen, die schnell arbeiten und einen gewissen Grad an Genauigkeit garantieren. Wir wollen zunächst die wohl simpelste Heuristik untersuchen, die man für dieses Problem angeben kann.

(8.24) Algorithmus (Die List-Heuristik). Gegeben sei ein Parallel-Shop-Problem durch $m, n, t_1, t_2, \ldots, t_n$.

1. **Initialisierung:**
   Belege die Maschinen $M_1, M_2, \ldots, M_m$ mit den Aufgaben $T_1, T_2, \ldots, T_m$.
   Setze $i := m + 1$.

2. Hat eine der Maschinen M_j ihre gegenwärtige Aufgabe erledigt, so belege M_j mit der Aufgabe T_i.
   Setze $i := i + 1$ und wiederhole 2, bis alle Jobs erledigt sind. △

Für unser Beispiel (8.23) ergibt die List-Heuristik (8.24) den in Abb. 8.8 angegebenen Belegungsplan, der offenbar wiederum optimal ist.

Das günstige Ergebnis bei Beispiel (8.23) sollte jedoch nicht zu Fehlschlüssen führen. Es gibt schlechte Beispiele, die zu recht ungünstigen Belegungsbeispielen führen.

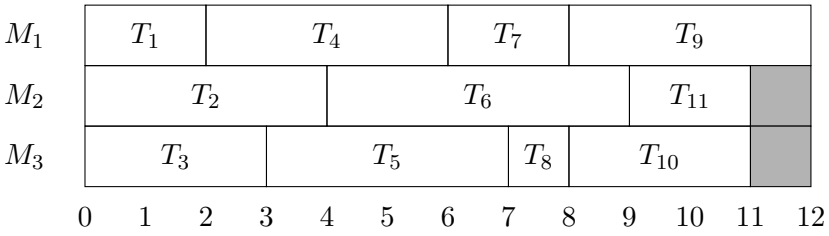


Abbildung 8.8: Belegungsplan der LIST-Heuristik für Beispiel (8.23)

(8.25) Beispiel.

$$m = 6,\ n = 11, \quad \underbrace{5, 5, 5, 5, 5}_{5\times},\ \underbrace{1, 1, 1, 1, 1}_{5\times},\ 6.$$

△

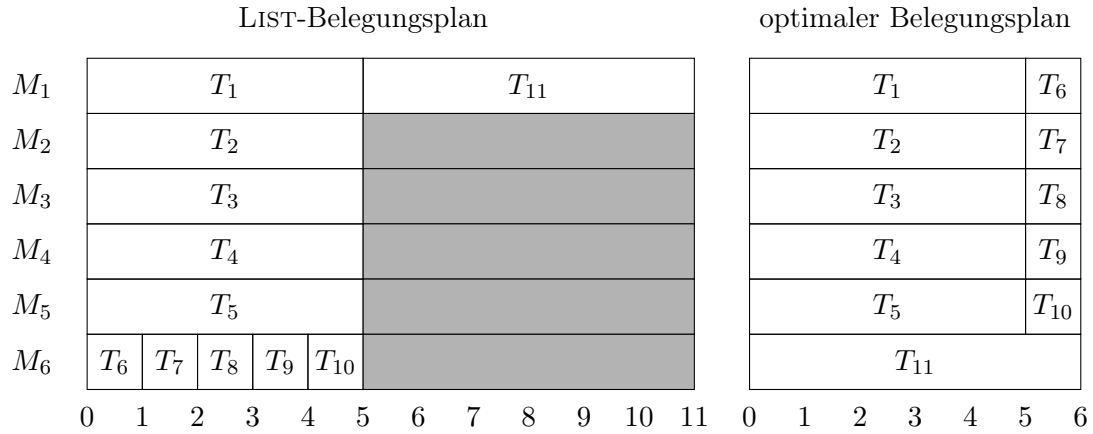


Abbildung 8.9: Schlechtes Beispiel für die LIST-Heuristik

Ganz allgemein kann man Beispiele angeben mit $n = 2m - 1$, bei denen die Fertigstellungszeit C_L der LIST-Heuristik $2m-1$ ist, während die optimale Fertigstellungszeit C_{opt} den Wert m hat. (Simple Modifikation von Beispiel (8.25).) Dies ist jedoch der schlechtest mögliche Fall wie der folgende Satz zeigt.

(8.26) Satz. *Gegeben sei ein Parallel-Shop-Problem mit m Maschinen und n Aufträgen. Sei C_L die Fertigstellungszeit der* LIST*-Heuristik und C_{opt} die optimale Fertigstellungszeit, dann gilt*

$$C_L \leq \Big(2 - \frac{1}{m}\Big) C_{\text{opt}}.$$

△

Der Beweis von Satz (8.26) folgt direkt aus dem nachfolgenden Lemma (8.27) zusammen mit der Überlegung, dass $C_{\text{opt}} \geq t$, wobei t die Bearbeitungszeit der Aufgabe ist, die als letzte im LIST-Belegungsplan ausgeführt wird.

(8.27) Lemma. *Ist t die Bearbeitungszeit der Aufgabe, die als letzte im* List*-Belegungsplan ausgeführt wird, dann gilt*

$$C_L \leq C_{\text{opt}}\Big(1 + \frac{(m-1)t}{m\,C_{\text{opt}}}\Big). \qquad \triangle$$

Beweis. Zum Zeitpunkt $C_L - t$ ist aufgrund der Definition der List-Heuristik keine Maschine leer. Eine beendet gerade ihre gegenwärtige Aufgabe und wird mit der Aufgabe der Länge t belegt. Es gilt also:

$$m\,C_{\text{opt}} \geq \sum_{i=1}^{n} t_i \quad \text{und} \quad \sum_{i=1}^{n} t_i - t \geq m\,(C_L - t),$$

und es folgt

$$\begin{aligned} m\,C_{\text{opt}} - t \geq m\,C_L - mt \;&\Longrightarrow\; C_L \leq C_{\text{opt}} + \frac{m-1}{m}t \\ &\Longrightarrow\; C_L \leq C_{\text{opt}}\Big(1 + \frac{(m-1)t}{m\,C_{\text{opt}}}\Big) \end{aligned} \qquad \square$$

Wir betrachten nun eine geringfügig modifizierte Version von List:

(8.28) Algorithmus (List Decreasing (LD)). Gegeben sei ein Parallel-Shop Problem $m, n, t_1, t_2, \ldots, t_n$.

1. Ordne die Aufgaben, so dass gilt $t_1 \geq t_2 \geq \cdots \geq t_m$.
2. Wende List an. $\triangle$

Ein Problem, bei dem List Decreasing ein relativ schlechtes Ergebnis liefert, ist das folgende.

(8.29) Beispiel.

$$m = 6,\ n = 13, \quad 11, 11, 10, 10, 9, 9, 8, 8, 7, 7, 6, 6, 6.$$

LD ergibt hier einen Belegungsplan mit $C_{\text{LD}} = 23$, während das Optimum 18 beträgt.

Optimaler Belegungsplan:	M_1 :	T_1, T_{10}
	M_2 :	T_2, T_9
	M_3 :	T_3, T_8
	M_4 :	T_4, T_7
	M_5 :	T_5, T_6
	M_6 :	T_{11}, T_{12}, T_{13}

$\triangle$

I. A. gibt es Beispiele mit $n = 2m+1$, $C_{\text{LD}} = 4m-1$ und $C_{\text{opt}} = 3m$. Jedoch sind diese die schlechtest möglichen Beispiele. Um dies zu zeigen, beginnen wir mit einem Hilfssatz.

(8.30) Lemma. *Gilt* $t_1 \geq t_2 \geq \ldots t_n > \frac{C_{\text{opt}}}{3}$*, dann ist* $C_{LD} = C_{\text{opt}}$. △

Beweis. Offenbar bearbeitet im optimalen Belegungsplan jede Maschine höchstens zwei Aufgaben (andernfalls gilt $3t_n > C_{\text{opt}}$). Zur Vereinfachung der Beweisführung führen wir $2m - n$ Aufgaben mit Bearbeitungszeit Null ein, so dass jede Maschine 2 Aufgaben erledigt, wobei wir die Bearbeitungsdauern der Aufgaben auf M_i mit a_i und b_i bezeichnen wollen und $a_i \geq b_i$ gelten soll. Also gilt

$$a_1 \geq a_2 \geq \cdots \geq a_m,$$
$$a_1 \geq b_1,\ \ldots,\ a_m \geq b_m.$$

Daraus folgt natürlich $a_1 = t_1$. Falls $a_i = t_i$ für $i = 1, \ldots, k-1$, aber $a_k < t_k$, dann gilt natürlich $t_k = b_i$ für ein $i \leq k-1$, denn $t_k > a_k \geq a_{k+1} \cdots \geq a_m$ und $a_j \geq b_j$ für $j = k, \ldots, n$. Tauschen wir nun b_i und a_k aus, so erhalten wir wiederum einen optimalen Belegungsplan, denn $C_{\text{opt}} \geq a_i + b_i > a_i + a_k \geq b_i + b_k$. Nach höchstens m Austauschoperationen dieser Art erhalten wir einen optimalen Belegungsplan mit

$$a_1 = t_1,\ a_2 = t_2,\ \ldots,\ a_m = t_m.$$

Analog können wir durch Austauschen erreichen, dass in einer optimalen Lösung $b_m = t_{m+1}$, $b_{m-1} = t_{m+2}$, $\ldots$, $b_1 = t_{2m}$ gilt. Dies ist aber eine Lösung, die LD liefert, und wir sind fertig. □

(8.31) Satz.

$$C_{LD} \leq \frac{1}{3}\Big(4 - \frac{1}{m}\Big)C_{\text{opt}}.$$ △

Beweis. Angenommen, die Behauptung stimmt nicht, dann gibt es ein Gegenbeispiel mit kleinstmöglichem n. Es gilt $t_1 \geq t_2 \geq \cdots \geq t_n$. Wir behaupten, dass T_n die letzte erledigte Aufgabe im LD-Belegplan ist. Wäre dies nicht so, dann könnten wir T_n aus unserem Beispiel entfernen, wobei die LD-Lösung des neuen Beispiels unverändert bliebe, die Optimallösung jedoch nicht schlechter wäre. Dies wäre ein Gegenbespiel zur Behauptung mit $n-1$ Aufgaben. Wenden wir nun Lemma (8.27) (mit $t = t_n$) an und benutzen wir die Annahme, dass wir mit einem Gegenbeispiel arbeiten, so erhalten wir:

$$\begin{aligned}
&\frac{1}{3}\Big(4 - \frac{1}{m}\Big)C_{\text{opt}} < C_{\text{LD}} \leq \Big(1 + \frac{(m-1)t_n}{m\,C_{\text{opt}}}\Big)C_{\text{opt}} \\
\Longrightarrow\quad & \frac{4}{3} - \frac{1}{3m} < \frac{C_{\text{LD}}}{C_{\text{opt}}} \leq 1 + \frac{m-1}{m}\frac{t_n}{C_{\text{opt}}} \\
\Longrightarrow\quad & \frac{1}{3}\frac{4m-1}{m} < \frac{(m-1)t_n}{m}\frac{1}{C_{\text{opt}}} \\
\Longrightarrow\quad & \frac{1}{3}C_{\text{opt}} < \frac{(m-1)t_n \cdot m}{(4m-1)m} \leq t_n.
\end{aligned}$$

Aus Lemma (8.30) folgt nun aber, dass $C_{\text{LD}} = C_{\text{opt}}$ gelten muss, ein Widerspruch. □

Maschinenbelegung mit abhängigen Aufgaben

Wir betrachten wiederum ein Parallel-Shop-Problem wie im letzten Abschnitt, das durch die Anzahl m der Maschinen, die Anzahl n der Aufgaben $T_1, \ldots, T_n$ und die Bearbeitungszeiten $t_1, \ldots, t_n$ beschrieben ist:

$$n, m,\ t_1, \ldots, t_n.$$

Eine zusätzliche Komplikation wird dadurch erzeugt, dass gewisse Aufgaben nicht unabhängig voneinander sind, d. h. dass eine Aufgabe T_i erst in Angriff genommen werden kann, wenn eine andere Aufgabe T_j vollständig erledigt ist. Solche Beschränkungen nennt man *Reihenfolgebedingungen*.

Schreiben wir symbolisch $T_j \longrightarrow T_i$ dafür, dass Aufgabe T_i erst beginnen kann, wenn Aufgabe T_j beendet ist, so kann man alle Reihenfolgebedingungen offenbar in einem *Reihenfolgedigraphen* darstellen. Ein solcher ist in Abbildung 8.10 angegeben.

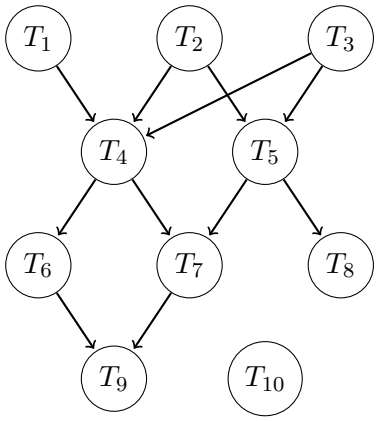


Abbildung 8.10: Reihenfolgedigraph

Bei einem Reihenfolgedigraphen sind nur unmittelbare Reihenfolgebedingungen dargestellt, z. B. $T_1 \longrightarrow T_4 \longrightarrow T_6 \longrightarrow T_9$. Daraus folgt natürlich, dass $T_1 \longrightarrow T_9$ gilt, obwohl dies im Diagramm nicht dargestellt ist.

Fügen wir zu einem Reihenfolgedigraphen (bzw. zu einem allgemeinen Digraphen) alle implizierten Reihenfolgebedingungen hinzu, so nennt man diesen Digraphen den *transitiven Abschluss* des Reihenfolgedigraphen (bzw. Digraphen).

Natürlich ist das Parallel-Shop-Problem mit Reihenfolgebedingungen ebenfalls $\mathcal{NP}$-schwer, und wir zeigen nun, dass die LIST-Heuristik auch für diesen Problemtyp gute Ergebnisse liefert.

(8.32) Algorithmus (Die LIST-Heuristik bei Reihenfolgebedingungen). Gegeben sei ein Parallel-Shop Problem durch

$$m, n,\ t_1, \ldots, t_n$$

und durch einen Reihenfolgedigraphen D.

1. Belege die Maschine M_1 mit der ersten verfügbaren Aufgabe, sagen wir T_1.
2. Hat eine Maschine ihre gegenwärtige Aufgabe erledigt, so weise ihr die nächste verfügbare Aufgabe zu. Falls aufgrund der Reihenfolgebedingungen keine Aufgabe zur Verfügung steht, bleibt die Maschine leer, bis die nächste Aufgabe freigegeben wird. △

(8.33) Beispiel. Betrachten wir das Parallel-Shop-Problem

$$3, 10, \quad 1, 2, 1, 2, 1, 1, 3, 6, 2, 1$$ △

mit dem Reihenfolgedigraphen aus Abbildung 8.10, so erhalten wir als LIST-Lösung den in Abbildung 8.11 angegebenen Belegungsplan mit $C_L = 9$.

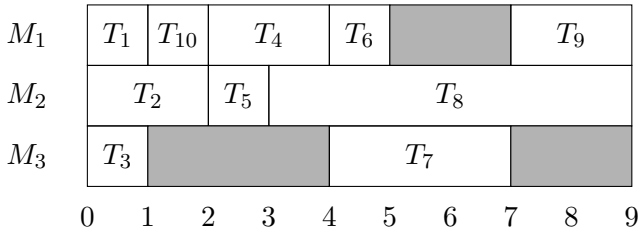


Abbildung 8.11: LIST-Lösung zu Beispiel (8.33)

Es ist erstaunlich, dass die Schranke aus Satz (8.26) auch für diesen komplizierteren Fall gilt.

(8.34) Satz. *Gegeben sei ein Parallel-Shop-Problem mit Reihenfolgebedingungen. Sei C_L die Fertigstellungszeit der* LIST*-Heuristik und C_{opt} die optimale Fertigstellungszeit. Dann gilt*

$$C_L \leq \Big(2 - \frac{1}{m}\Big) C_{\text{opt}}.$$ △

Beweis. Wir bezeichnen mit T_1^* eine Aufgabe, die im LIST-Belegplan als letzte endet. Sei k die größte ganze Zahl, für die es eine Menge von Aufgaben

$$T_k^* \longrightarrow T_{k-1}^* \longrightarrow \ldots \longrightarrow T_1^*$$

gibt, so dass die folgende Eigenschaft erfüllt ist:

> Falls zu irgendeinem Zeitpunkt zwischen der Startzeit von T_k^* und der Endzeit C_L von T_1^* eine Maschine leer ist, dann wird irgendeine der Aufgaben T_i^* bearbeitet.

Es ist offensichtlich, dass es überhaupt so eine Folge $T_k^* \longrightarrow \ldots \longrightarrow T_1^*$ gibt. Der Hauptschritt des Beweises ist der Beweis der folgenden Behauptung.

> Zu jedem Zeitpunkt vor dem Beginn der Ausführung von T_k^* sind alle m Maschinen belegt.

Nehmen wir an, dass diese Behauptung falsch ist. Dann finden wir einen letzten Zeitpunkt vor dem Beginn von T_k^*, zu dem eine Maschine leer steht. Warum bearbeitet diese Maschine nicht T_k^*? Die einzige mögliche Antwort ist, dass es eine Aufgabe T_j gibt mit $T_j \longrightarrow T_k^*$, die noch nicht beendet ist. Setzten wir nun $T_{k+1}^* := T_j$, so haben wir eine längere Kette von Aufgaben mit den gewünschten Eigenschaften. Ein Widerspruch zur Maximalität von k.

Sei nun w_i die gesamte Leerzeit der Maschine M_i, für $i = 1, \ldots, m$, dann gilt

$$\sum_{j=1}^{n} t_j + \sum_{i=1}^{m} w_i = m\, C_L.$$

Sei t die Gesamtausführungszeit der Aufgaben $T_k^*, \ldots, T_1^*$, dann implizieren die beiden Eigenschaften der Kette $T_k^* \longrightarrow \ldots \longrightarrow T_1^*$:

$$\sum_{i=1}^{m} w_i \leq (m-1)\, t.$$

(Die Leerzeiten fallen erst nach Beginn von T_k^* an, dann ist aber immer eine Maschine beschäftigt.) Also erhalten wir

$$m\, C_L \leq \sum_{j=1}^{n} t_j + (m-1)\, t \leq m\, C_{\text{opt}} + (m-1)\, C_{\text{opt}} \implies C_L \leq \Big(1 + \frac{m-1}{m}\Big) C_{\text{opt}}.$$

□

Leider ist kein zu Satz (8.31) analoges Resultat für Probleme mit Reihenfolgebedingungen bekannt.

Forschungsproblem: Welche Gütegarantie hat die List Decreasing Heuristik für das Parallel-Shop-Problem mit Reihenfolgebedingungen?

Das Packen von Kisten (Bin-Packing)

Das Problem, das wir nun behandeln wollen, ist in gewissem Sinne dual zum Parallel-Shop-Problem mit unabhängigen Aufgaben. Bei letzterem haben wir Aufgaben und Maschinen gegeben und suchen eine möglichst kurze Gesamtfertigstellungszeit. Beim *Bin-Packing-Problem* hat man dagegen eine Anzahl von Aufgaben und ihre jeweilige Dauer und eine späteste Fertigstellungszeit vorgegeben. Gesucht ist eine minimale Zahl von Maschinen, um die Aufgaben in der vorgegebenen Zeit zu erledigen.

Es ist üblich, dieses Problem als eindimensionales Kistenpackungsproblem zu beschreiben. Jede Kiste habe eine Höhe d und jeder Gegenstand eine Höhe t_i (natürlich kann man auch Gewichte etc. wählen). Die Form der Grundfläche ist bei allen Gegenständen identisch. Das Ziel ist, eine minimale Zahl von Kisten zu finden, die alle Gegenstände aufnehmen. Für dieses Problem betrachten wir die folgende Heuristik.

(8.35) Algorithmus (First-Fit (FF)). Gegeben sei ein Bin-Packing-Problem durch die Folge $d, t_1, \ldots, t_n$. Die Gegenstände werden in der Reihenfolge, wie sie in der Liste vorkommen, behandelt. Jeder Gegenstand wird in die erste Kiste, in die er passt, gelegt.△

(8.36) Beispiel. Gegeben sei ein Bin-Packing-Problem durch die Kistenhöhe $d = 101$ und die folgenden 37 Gegenstände: 6 (7×), 10 (7×), 16 (3×), 34 (10×), 51 (10×). Die FF-Lösung ist in Abbildung 11.7 dargestellt.

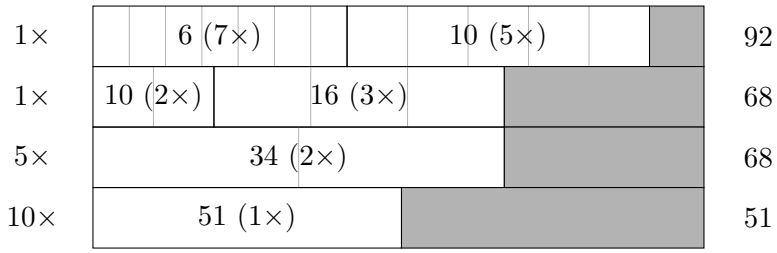


Abbildung 8.12: FF-Lösung zu Beispiel (8.36)

Das heißt, bei der FF-Lösung werden 17 Kisten benötigt. Die optimale Lösung benötigt nur 10 Kisten, wie Abbildung 8.13 zeigt.

Abbildung 8.13: Optimallösung zu Beispiel (8.36)

Daraus folgt, dass $\frac{m_{\text{FF}}}{m_{\text{opt}}}$ (m_{FF} ist die durch FF bestimmte Kistenzahl und m_{opt} die optimale Kistenzahl m) durchaus $\frac{17}{10}$ sein kann. Die folgende Abschätzung

$$m_{\text{FF}} \leq 1 + 2m_{\text{opt}}$$

ist trivial, denn die FF-Lösung füllt jede benutzte Kiste (außer vielleicht der letzten) im Durchschnitt mindestens bis zur Hälfte, also gilt:

$$\frac{d}{2}(m_{FF} - 1) \leq \sum_{j=1}^{n} t_j \leq m_{\text{opt}} \cdot d.$$

Es ist erheblich schwieriger zu zeigen, dass der im Beispiel (8.36) gefundene Bruch $\frac{17}{10}$ asymptotisch der beste ist.

(8.37) Satz. *Gegeben sei ein Bin-Packing-Problem. m_{FF} sei die durch* First Fit *gefundene Kistenzahl, m_{opt} die optimale, dann gilt*

$$m_{FF} < \frac{17}{10} m_{\text{opt}} + 2.$$

△

Im Beweis von Satz (8.37) benutzen wir die Existenz einer Funktion

$$w : [0,1] \to [0,1],$$

die die folgenden Eigenschaften hat:

Eigenschaft 1:

$$\frac{1}{2} < t \leq 1 \implies w(t) = 1.$$

Eigenschaft 2:

$$\left.\begin{array}{r} k \geq 2 \\ t_1 \geq t_2 \geq \ldots \geq t_k \\ \sum_{i=1}^{k} t_i \leq 1 \\ \sum_{i=1}^{k} w(t_i) = 1 - s,\ s > 0 \end{array}\right\} \implies \sum_{i=1}^{k} t_i \leq 1 - t_k - \frac{5}{9}s.$$

Eigenschaft 3:

$$\sum_{i=1}^{k} t_i \leq 1 \implies \sum_{i=1}^{k} w(t_i) \leq \frac{17}{10}.$$

Wir werden später die Existenz einer solchen Funktion beweisen und nehmen zum Beweis von (8.37) zunächst die Existenz von w an.

Beweis (von Satz (8.37)). Wir nehmen an, dass alle Kisten die Höhe 1 und die Gegenstände eine Höhe von $t'_i = \frac{t_i}{d}$ haben ($1 \leq i \leq n$). Wir nehmen zusätzlich an, dass jeder Gegenstand i ein *Gewicht* $w(t'_i)$ hat. Das Gewicht einer Kiste sei die Summe der Gewichte der in sie gelegten Gegenstände.

Betrachten wir die FF-Lösung, so bezeichnen wir mit $B_1^*, \ldots, B_m^*$ diejenigen Kisten (in ihrer Originalreihenfolge), die ein Gewicht kleiner als 1 haben, und zwar habe B_i^* das Gewicht $1 - s_i$, $s_i > 0$. Die ungenutzte Höhe von B_i^* bezeichnen wir mit c_i. Wir setzen $c_0 = 0$ und behaupten

$$c_i \geq \frac{5}{9}s_i + c_{i-1} \quad \text{für} \quad i = 1, 2, \ldots, m-1. \tag{1}$$

Um (1) zu zeigen, stellen wir zunächst fest, dass jeder Gegenstand x, der in einer der Kisten B_i^* liegt, höchstens eine Höhe von $\frac{1}{2}$ hat, denn andernfalls wäre nach Eigenschaft 1 sein Gewicht 1 und somit hätte B_i^* ein Gewicht nicht kleiner als 1.

Daraus folgt $c_i < \frac{1}{2}$ für $i = 1, \ldots, m-1$, denn andernfalls hätte ein Gegenstand aus B_m^*, der auch höchstens die Höhe $\frac{1}{2}$ hat, in die Kiste B_i^* gelegt werden können.

Daraus können wir schließen, dass jede Kiste B_i^* mindestens 2 Gegenstände enthält, denn es gilt $c_i < \frac{1}{2}$, und jeder Gegenstand in B_i^* hat höchstens die Höhe $\frac{1}{2}$.

Sei nun B_i^* irgendeine Kiste mit $1 \leq i \leq m-1$, und nehmen wir an, dass sie mit Objekten der Höhe $x_1 \geq x_2 \geq \cdots \geq x_k$ gefüllt ist. Wir wissen bereits, dass $k \geq 2$ gilt, und somit folgt aus Eigenschaft 2:

$$1 - c_i = \sum_{i=1}^{k} x_i \leq 1 - x_k - \frac{5}{9} s_i \implies c_i \geq \frac{5}{9} s_i + x_k.$$

Da $x_k > c_{i-1}$ (andernfalls wäre x_k in der Kiste B_{i-1}^*), gilt (1). Nun gilt aufgrund von (1)

$$\sum_{i=1}^{m-1} s_i \leq \frac{9}{5} \sum_{i=1}^{m-1} (c_i - c_{i-1}) = \frac{9}{5} \underbrace{c_{m-1}}_{<1/2} < \frac{9}{10}.$$

Es folgt $\sum_{i=1}^{m} s_i \leq \frac{9}{10} + \underbrace{s_m}_{\leq 1} < 2$. Die letzte Gleichung impliziert

$$\begin{aligned}
\sum_{j=1}^{n} w(t_j') &= \sum_{i=1}^{m} \sum_{j \in B_i^*} w(t_j') + \sum_{j \notin B_i^*} w(t_j') \\
&\geq \sum_{i=1}^{m} (1 - s_i) + m' \\
&= m + m' - \sum_{i=1}^{m} s_i \\
&> m_{FF} - 2.
\end{aligned}$$

Andererseits impliziert Eigenschaft 3

$$\sum_{j=1}^{n} w(t_j') = \sum_{i=1}^{m_{\text{opt}}} \sum_{j \in B_i^{\text{opt}}} w(t_j') \leq m_{\text{opt}} \cdot \frac{17}{10}.$$

Es gilt $\sum_{j \in B_i^{\text{opt}}} t_j \leq 1 \implies \sum_{j \in B_i^{\text{opt}}} w(t_j') \leq \frac{17}{10}$. Insgesamt gilt also

$$m_{FF} \leq \sum_{j=1}^{n} w(t_j') + 2 \leq 2 + \frac{17}{10} m_{\text{opt}}.$$

□

Um den Beweis von Satz (8.37) zu vervollständigen, müssen wir noch die Existenz der Funktion mit den angegebenen Eigenschaften nachweisen. Wir geben diese Funktion an:

$$w(x) := \begin{cases} \frac{6}{5}x & \text{falls } 0 \leq x < \frac{1}{6} \\ \frac{9}{5}x - \frac{1}{10} & \text{falls } \frac{1}{6} \leq x < \frac{1}{3} \\ \frac{6}{5}x + \frac{1}{10} & \text{falls } \frac{1}{3} \leq x \leq \frac{1}{2} \\ 1 & \text{falls } \frac{1}{2} < x \leq 1 \end{cases}$$

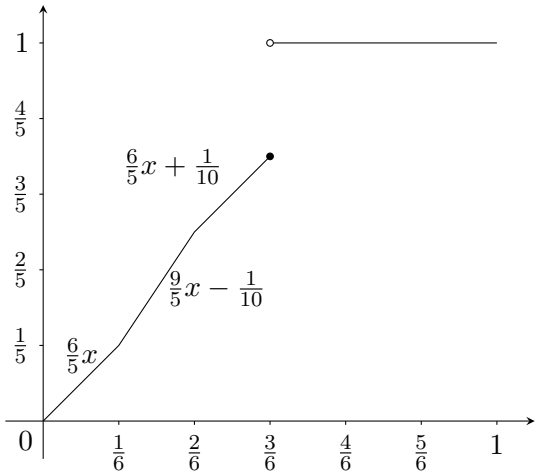


Abbildung 8.14: Graph von $w(x)$

w hat offenbar die Eigenschaft 1. Der Beweis der übrigen Eigenschaften ist recht kompliziert, wir spalten die Kette der Argumente in mehrere Behauptungen auf.

Behauptung 1: $\frac{1}{3} \leq x \leq \frac{1}{2} \implies w\big(\frac{1}{3}\big) + w\big(x - \frac{1}{3}\big) = w(x)$, trivial.

Behauptung 2: $0 \leq x, y \leq \frac{1}{6} \implies w(x) + w(y) \leq w(x + y)$, trivial.

Behauptung 3: $0 \leq x \leq \frac{1}{2} \implies \frac{6}{5}x \leq w(x) \leq \frac{3}{2}x$, trivial

Behauptung 4: $x \leq x^* \leq \frac{1}{2} \implies w^* - w(x) \leq \frac{9}{5}(x^* - x)$, trivial.

Behauptung 5: Es gilt

$$\left.\begin{array}{c} k \geq 2 \\ x_1 \geq x_2 \geq \cdots \geq x_k > 0 \\ 1 - x_k \leq \sum_{i=1}^{k} x_i \leq 1 \end{array}\right\} \implies \sum_{i=1}^{k} w(x_j) \geq 1.$$

Beweis. Wir können Folgendes annehmen:

$x_1 \leq \frac{1}{2}$, denn sonst wäre $w(x_1) = 1$.

$x_2 < \frac{1}{3}$, denn sonst wäre $w(x_1) + w(x_2) \geq 2w(\frac{1}{3}) = 1$.

$x_k > \frac{1}{6}$, denn sonst wäre $\sum_{i=1}^{k} x_i \geq \frac{5}{6}$, und aus Behauptung 3 würde folgen

$$\sum_{i=1}^{k} w(x_i) \geq \sum_{i=1}^{k} \frac{6}{5} x_i = \frac{6}{5} \sum_{i=1}^{k} x_i \geq 1.$$

Wir betrachten zunächst den Fall $k = 2$. $x_1 + x_2 \geq 1 - x_2 \implies x_1 \geq 1 - 2x_2 \implies x_1 \geq \frac{1}{3}$, $x_2 \geq \frac{1-x_1}{2}$. Und somit

$$w(x_1) + w(x_2) = \underbrace{\frac{6}{5}x_1 + \frac{1}{10}}_{w(x_1)} + \underbrace{\frac{9}{5}x_2 - \frac{1}{10}}_{w(x_2)} \geq \frac{6}{5}x_1 + \frac{9}{10}(1 - x_1) \geq 1.$$

Nun der Fall $k \geq 3$. Falls $x_1 \geq \frac{1}{3}$, dann gilt

$$\begin{aligned}\sum_{i=1}^{k} w(x_i) &\geq \overbrace{\frac{6}{5}x_1 + \frac{1}{10}}^{w(x_1)} + \overbrace{\frac{9}{5}x_2 - \frac{1}{10}}^{w(x_2)} + \overbrace{\frac{6}{5}\sum_{i=3}^{k} x_i}^{\substack{\text{Abschätzung} \\ \text{nach Beh. 3}}} = \frac{6}{5}\sum_{i=1}^{k} x_i + \frac{3}{5}x_2 \\ &\geq \frac{6}{5}(1 - x_k) + \frac{3}{5}x_k = \frac{6}{5} - \frac{3}{5}\underbrace{x_k}_{<1/3} \geq 1.\end{aligned}$$

Falls $x_1 < \frac{1}{3}$, dann gilt

$$\begin{aligned}\sum_{i=1}^{k} w(x_i) &\geq \overbrace{\frac{9}{5}(x_1 + x_2) - \frac{1}{5}}^{w(x_1)+w(x_2)} + \overbrace{\frac{6}{5}(x_3 + \cdots + x_k)}^{\text{Absch. nach Beh. 3}} \\ &= \frac{6}{5}\sum_{i=1}^{k} x_i + \frac{3}{5}(x_1 + x_2) - \frac{1}{5} \geq \frac{6}{5}(1 - x_k) + \frac{6}{5}x_k - \frac{1}{5} = 1.\end{aligned}$$

□

Behauptung 6: (Eigenschaft 2) Es gilt

$$\left.\begin{array}{c} k \geq 2 \\ x_1 \geq x_2 \geq \cdots \geq x_k > 0 \\ \sum_{i=1}^{k} x_i \leq 1 \\ \sum_{i=1}^{k} w(x_i) = 1 - s,\ s > 0 \end{array}\right\} \implies \sum_{i=1}^{k} x_i \leq 1 - x_k - \frac{5}{9}s.$$

Beweis. Wir verneinen Behauptung 5. Ist also $\sum w(x_i) < 1$, so muss eine der vier Voraussetzungen von Behauptung 5 nicht gelten. Da wir in Behauptung 6 drei der vier Voraussetzungen fordern, muss $1 - x_k \leq \sum x_i$ verletzt sein. Wir definieren $t := 1 - x_k - \sum_{i=1}^{k} x_i$, und nach obiger Überlegung gilt $t > 0$. Folglich gilt $x_1 + x_2 + t \leq 1 - x_k < 1$ und deshalb gibt es Zahlen x_1^*, x_2^*, so dass

$$x_1 \leq x_1^* \leq \frac{1}{2}, \quad x_2 \leq x_2^* \leq \frac{1}{2}, \quad x_1^* + x_2^* = x_1 + x_2 + t.$$

Setzen wir $x_i^* = x_i$, $i = 3, \ldots, k$, so sind die Voraussetzungen von Behauptung 5 erfüllt, und wir erhalten $\sum_{i=1}^{k} w(x_i^*) \geq 1$. Also gilt

$$w(x_1^*) + w(x_2^*) \geq w(x_1) + w(x_2) + s.$$

Behauptung 4 impliziert nun

$$s \leq w(x_1^*) - w(x_1) + w(x_2^*) - w(x_2) \leq \frac{9}{5}(x_1^* - x_1) + \frac{9}{5}(x_2^* - x_2) = \frac{9}{5}t.$$

Also folgt $t \geq \frac{5}{9}s$, und damit gilt

$$\sum_{i=1}^{k} x_i = 1 - x_k - t \leq 1 - x_k - \frac{5}{9}s.$$

□

Behauptung 7:

$$\sum_{i=1}^{k} x_i \leq \frac{1}{2} \implies \sum_{i=1}^{k} w(x_i) \leq \frac{7}{10}.$$

Beweis. Aufgrund von Behauptung 1 können wir annehmen, dass $x_i \leq \frac{1}{3}$ gilt für $i = 1, \ldots, k$ (andernfalls könnten wir jedes $x_i > \frac{1}{3}$ ersetzen durch $x' = \frac{1}{3}$ und $x_i'' = x_i - \frac{1}{3}$).

Aufgrund von Behauptung 2 können wir annehmen, dass höchstens ein x_i kleiner gleich $\frac{1}{6}$ ist (andernfalls könnten wir $x_i, x_j \leq \frac{1}{6}$ durch $x_{ij} := x_i + x_j$ ersetzen). Wir unterscheiden vier Fälle:

i) $k = 1$ und $x_1 \leq \frac{1}{3}$, dann gilt: $w(x_1) \leq w(\frac{1}{3}) = \frac{1}{2} < \frac{7}{10}$.

ii) $k = 2$, $\frac{1}{6} \leq x_2 \leq x_1 \leq \frac{1}{3}$, dann gilt: $w(x_1) + w(x_2) = \frac{9}{5}(x_1 + x_2) - \frac{2}{10} \leq \frac{7}{10}$.

iii) $k = 2$, $x_2 \leq \frac{1}{6} \leq x_1 \leq \frac{1}{3}$, dann gilt: $w(x_1) + w(x_2) \leq w(\frac{1}{3}) + w(\frac{1}{6}) = \frac{7}{10}$.

iv) $k = 3$, $x_3 \leq \frac{1}{6} \leq x_2 \leq x_1 \leq \frac{1}{3}$, dann gilt: $w(x_1) + w(x_2) + w(x_3) = \frac{9}{5}(x_1 + x_2) - \frac{2}{10} + \frac{6}{5}x_3 \leq \frac{9}{5}(x_1 + x_2 + x_3) - \frac{2}{10} \leq \frac{7}{10}$.

□

Behauptung 8: (Eigenschaft 3)

$$\sum_{i=1}^{k} x_i \leq 1 \implies \sum_{i=1}^{k} w(x_i) \leq \frac{17}{10}.$$

Beweis. Falls $x_i \leq \frac{1}{2}$ für alle i, dann impliziert Behauptung 3

$$\sum_{i=1}^{k} w(x_i) \leq \frac{3}{2} \sum_{i=1}^{k} x_i \leq \frac{3}{2} < \frac{17}{10}.$$

Ist eines der x_i größer als $\frac{1}{2}$, sagen wir $x_1 > \frac{1}{2}$, dann folgt aus Behauptung 7 ($\sum_{i=2}^{k} x_i \leq \frac{1}{2}$)

$$\sum_{i=2}^{k} w(x_i) \leq \frac{7}{10},$$

und somit $w(x_1) + \sum_{i=2}^{k} w(x_i) \leq \frac{17}{10}$. □

Damit ist unsere Analyse der FIRST-FIT Heuristik beendet. Eine mögliche Verbesserung der Methode liegt auf der Hand.

(8.38) Algorithmus (FIRST-FIT DECREASING (FFD)). Gegeben sei ein Bin-Packing-Problem durch $d, t_1, \ldots, t_n$.

1. Ordne zunächst alle t_i, so dass gilt $t_1 \geq t_2 \geq \cdots \geq t_n$.
2. Wende FF an. △

Wir betrachten das folgende Beispiel.

(8.39) Beispiel. Sei

$$d = 60,\ 31\,(6\times),\ 17\,(6\times),\ 16\,(6\times),\ 13\,(12\times),$$

so benötigt FFD 11 Kisten, während in der optimalen Lösung nur 9 Kisten benutzt werden.

$$\underbrace{\begin{array}{ll} 6\times & 31, 17 \\ 2\times & 16, 16, 16 \\ 3\times & 13, 13, 13, 13 \end{array}}_{\text{FFD}} \qquad \underbrace{\begin{array}{ll} 6\times & 31, 16, 13 \\ 3\times & 17, 17, 13, 13 \end{array}}_{\text{Optimum}}$$ △

Ist m_{FFD} die Kistenzahl der durch FFD gefundenen Lösungen und m_{opt} die optimale Kistenzahl, so kann gezeigt werden:

(8.40) Satz.

$$m_{FFD} < 4 + \frac{11}{9}\, m_{\text{opt}}.$$ △

Der Beweis ist in seiner Struktur ähnlich wie der von Satz (8.37). Es treten jedoch erheblich kompliziertere technische Detailprobleme auf. Der erste Beweis für Satz (8.40) benötigte rund 70 Seiten; ein kürzerer ist zu finden in Baker (1985). Dósa (2007) hat bewiesen, dass

$$m_{\text{FFD}} \leq \frac{2}{3} + \frac{11}{9}\, m_{opt}$$

die bestmögliche Schranke für FFD ist, d. h. es gibt Beispiele, bei denen dieser Fehler tatsächlich erreicht wird.

Zum Abschluss der Analyse von FFD möchten wir noch bemerken, dass FFD (wie auch einige andere Heuristiken) eine paradoxe Eigenschaft hat. Es gilt nämlich, dass die Entfernung eines Gegenstandes die Kistenzahl erhöhen kann.

(8.41) Beispiel. Gegeben sei ein Bin-Packing-Problem durch:

$$d = 396,\ 285,\ 188\,(6\times),\ 126\,(18\times),\ 115\,(3\times), 112\,(3\times),$$
$$75,\ 60,\ 51,\ 12\,(3\times),\ 10\,(6\times),\ 9\,(12\times).$$

Es werden bei der FFD-Lösung 12 Kisten benötigt. Diese Lösung füllt alle Behälter vollständig und ist optimal. Entfernen wir den Gegenstand mit Höhe 75, so gilt $m_{\mathrm{FFD}} = 13$. Der letzte Gegenstand der Höhe 9 muss in einen weiteren Behälter gelegt werden. △

Zwischen FF und FFD besteht (für praktische Zwecke) ein fundamentaler Unterschied. Um FFD anwenden zu können, müssen vor Beginn der Festlegung des Belegplans alle Zahlen t_i bekannt sein. Das ist bei FF nicht der Fall, hier wird jede gerade verfügbare Zahl verarbeitet, und für den Algorithmus ist es uninteressant, welche und wieviele Zahlen t_i später noch erscheinen. FF ist also ein Online-Algorithmus, FFD ist das nicht.

Das Bin Packing Problem ist sicherlich nicht das anwendungsreichste kombinatorische Optimierungsproblem, aber es ist eine beliebte „Spielwiese“ der Heuristik-Designer. Die hier angegebenen Heuristiken FF und FFD sind nicht die besten bezüglich des (beweisbaren) Worst-Case-Verhaltens. Es gibt polynomiale Verfahren, die für jedes feste ε eine Lösung garantieren, die nicht schlechter als das $(1+\varepsilon)$-fache des Wertes der Optimallösung ist. Diese Verfahren sind allerdings recht kompliziert und basieren auf der Ellipsoidmethode. Die Literatur zum Bin-Packing-Thema ist außerordentlich umfangreich. Der Aufsatz Coffman et al. (1996) z. B. gibt einen Überblick über die bisherige Entwicklung.

Online-Optimierung

Es bietet sich an, dieses Kapitel über Bin-Packing mit einem Exkurs über Online-Algorithmen abzuschließen. Wir beginnen mit einer einfachen Beobachtung.

(8.42) Satz. *Kein Online-Algorithmus für das Bin-Packing-Problem hat eine Gütegarantie, die kleiner als $\frac{4}{3}$ ist.* △

Beweis. Wir nehmen an, dass der Algorithmus B ein Online-Algorithmus mit Gütegarantie $< \frac{4}{3}$ ist.

O. B. d. A. können wir annehmen, dass $d = 2$ gilt. Wir konstruieren nun zwei Eingabefolgen. Die erste Folge, genannt F_1, besteht aus $2k$, $k \geq 2$, Elementen mit Höhe $1-\varepsilon$, wobei ε eine sehr kleine rationale Zahl ist. Offenbar ist $m_{\mathrm{opt}}(F_1) = k$. Was B genau macht, wissen wir nicht. Aber B kann in jeden Behälter höchstens 2 Gegenstände packen. Wir bezeichnen nach Ausführung von B mit Eingabefolge F_1 mit b_i die Anzahl der Behälter mit i Gegenständen $(i = 1, 2)$. Dann gilt $m_B(F_1) = b_1 + b_2$, $b_1 + 2b_2 = 2k$ und somit $m_B(F_1) = 2k - b_2$. Da wir angenommen haben, dass B eine Gütegarantie kleiner als $\frac{4}{3}$ hat, ergibt sich für die Folge F_1: $m_B(F_1) = 2k - b_2 < \frac{4}{3} m_{opt}(F_1) = \frac{4}{3}k$, und somit $b_2 > \frac{2}{3}k$, also eine untere Schranke für b_2.

Unsere zweite Eingabefolge F_2 besteht aus $4k$ Gegenständen, wobei die ersten $2k$ Gegenstände die Höhe $1-\varepsilon$ und die folgenden $2k$ Gegenstände die Höhe $1+\varepsilon$ haben. Offensichtlich gilt $m_{opt}(F_2) = 2k$. Da der Online-Algorithmus bei Anwendung auf F_2

nichts davon weiß, dass nach den ersten $2k$ Elementen noch $2k$ Gegenstände kommen, verhält er sich auf den ersten $2k$ Elementen genau so wie bei der Verarbeitung von F_1. Danach hat er keine Wahl mehr. Das Bestmögliche, was B noch erreichen kann, ist, b_1 der restlichen $2k$ Elemente mit Höhe $1+\varepsilon$ auf die Behälter zu verteilen, in denen bisher nur ein Gegenstand ist (wenn B sich „dumm“ verhält, kann B p dieser b_1 Elemente in eigene Behälter legen), und die übrigen $2k-b_1$ Elemente jeweils einzeln in einen Behälter zu legen. Daraus ergibt sich

$$m_B(F_2) = m_B(F_1) + 2k - b_1 + p = (2k - b_2) + (2k - (2k - b_2)) + p = 2k + b_2 + p.$$

Die Eingabefolge F_2 liefert somit $m_B(F_2) = 2k + b_2 + p < \frac{4}{3} m_{\text{opt}}(F_2) = \frac{4}{3} 2k$, woraus $b_2 < \frac{2}{3}k$ folgt. Dies ist ein Widerspruch, und damit ist unsere Annahme falsch, dass der Online-Algorithmus B eine Gütegarantie kleiner als $\frac{4}{3}$ hat. □

Bei Online-Algorithmen hat es sich eingebürgert, nicht von Gütegarantien zu sprechen, sondern die Qualität mit *Wettbewerbsfähigkeit* (engl. competitiveness) zu bezeichnen. Bei genauer Betrachtung müssen wir die Eingabekonventionen ein wenig modifizieren. Haben wir ein Optimierungsproblem Π gegeben, und ist $I \in \Pi$ ein Problembeispiel, so gehen wir davon aus, dass die Eingabedaten als Folge $s = (s_1, \ldots, s_k)$ auftreten und dass die Daten in der Reihenfolge ihres Auftretens abgearbeitet werden. Bei (normalen) Optimierungsproblemen ist die Reihenfolge der Daten irrelevant, bei Online-Optimierungsproblemen kann das Ergebnis der Ausführung eines Online-Algorithmus sehr stark von der Datenfolge abhängen. Für eine (normale) Probleminstanz I gibt es also durch Permutation der Daten sehr viele Online-Varianten. Dies ist bei der folgenden Definition zu beachten.

(8.43) Definition. *Sei Π ein Online-Minimierungsproblem (die Definition für Online-Maximierungsprobleme ist analog). Für ein Problembeispiel $I \in \Pi$ bezeichnen wir mit $C_{\text{opt}}(I)$ den Optimalwert von I und mit $C_A(I)$ den Wert, den ein Online-Algorithmus A bei Anwendung von A auf I berechnet. Gibt es Werte $c, b \in \mathbb{R}$ so dass für alle $I \in \Pi$ die folgende Abschätzung gilt*

$$c_A(I) \leq c \cdot C_{\text{opt}}(I) + b,$$

*so heißt der Algorithmus A c-*kompetitiv. △

Man beachte, dass hier keine Voraussetzungen über die Laufzeit von A gemacht werden. Es wird lediglich verlangt, dass der Algorithmus A eine Entscheidung fällt, sobald eine neue Information (beim Bin-Packing eine weitere Höhe t_i) auftritt. A kann dann (im Prinzip) beliebig lange rechnen. Ein neuer Eingabewert erscheint erst, nachdem A eine endgültige Entscheidung über die Verarbeitung der vorliegenden Information getroffen hat. Und diese Entscheidung darf im weiteren Ablauf des Verfahrens nicht mehr revidiert werden.

Natürlich gibt es unzählige Varianten dieses Konzepts. Spielt die Zeit zur Entscheidungsfindung eine wichtige Rolle, muss also der Online-Algorithmus innerhalb einer vorher festgelegten Zeitdauer eine Antwort finden, so spricht man von *Echtzeit-Algorithmen.* Was „Echtzeit“ ist, hängt stark von den Anwendungsszenarien ab. So ist z. B. klar, dass

bei Anwendungen in der Telekommunikation (Entscheidungen über den Verbindungsaufbau und den Datentransfer) erheblich schneller reagiert werden muss, als bei Problemen der Verkehrssteuerung.

Es gibt auch Anwendungsfälle, wo ein Online-Algorithmus nicht sofort reagieren muss. Es kann erlaubt sein, dass der Algorithmus eine gewisse Zeit wartet und die entstehenden Aufgaben „puffert“. Solche Fragestellungen treten häufig in der innerbetrieblichen Logistik auf, wo die Erledigung einzelner Transportaufgaben nicht unerheblichen Zeitaufwand erfordert und wo bei der Entscheidung, welche der anstehenden Aufgaben als nächste erledigt wird, bis zur Fertigstellung des laufenden Transportauftrags gewartet wird, um bei der Entscheidungsfindung alle inzwischen aufgelaufenen Aufgaben einzubeziehen. Es kann auch erlaubt sein, dass einmal getroffene Entscheidungen in beschränkter Weise revidiert werden können. Dies sind nur einige Beispiele von praxisrelevanten sinnvollen Modifikationen des hier betrachteten „Online-Konzepts“.

Einer Frage wollen wir am Ende dieses Abschnitts noch nachgehen: Ist die untere Schranke aus Satz (8.42) für die Kompetitivität von Online-Algorithmen für das Bin-Packing bestmöglich?

Um zu zeigen, dass das nicht der Fall ist, modifizieren wir den Beweis von (8.42) ein wenig. Die Beweismethode dürfte klar sein. Wir produzieren Eingabesequenzen, bei denen unser Gegner (der hypothetische Online-Algorithmus) in Fallen läuft, die zu einer schlechten Kompetitivität führen.

(8.44) Satz. *Die Kompetitivität von Online-Bin-Packing-Algorithmen ist nicht besser als* $1,5$. △

Beweis. Wir erzeugen ein Bin-Packing-Problem, mit Behälterhöhe 3, bei dem alle Gegenstände die Höhe 1 oder 2 haben. Im ersten Schritt bieten wir unserem Online-Algorithmus B einen Gegenstand der Höhe 1 an. Er muss diesen in einen Behälter legen. Dann bieten wir B einen zweiten Gegenstand der Höhe 1 an. Legt B diesen in einen zweiten Behälter, hören wir auf (B ist ja unbekannt, wieviele Objekte zu bearbeiten sind). In diesem Fall hat B zwei Behälter benötigt, während die Optimallösung nur einen braucht. Der Fehler ist also 100%.

Legt B das zweite Element in den Behälter mit dem ersten Gegenstand, bieten wir anschließend zwei Gegenstände der Höhe 2 an. Diese muss B in je einen neuen Behälter legen, B benötigt also 3 Behälter, die Optimallösung kommt mit zwei Behältern aus. Der Fehler beträgt also 50% und hieraus folgt die Behauptung. □

Die nächste Frage stellt sich von selbst: Wie weit kann man (mit etwas raffinierteren Eingangsfolgen) die untere Schranke nach oben treiben? Das Erstaunliche ist, dass der unvermeidbare Fehler von 50% aus Satz (8.44) fast optimal ist.

(8.45) Satz.

(a) *Der bestmögliche Kompetitivitätsfaktor für das Online-Bin-Packing ist nicht kleiner als* $c = 1,5401...$ *(siehe van Vliet (1992)).*

(b) *Es gibt einen c-kompetitiven Online-Bin-Packing-Algorithmus mit* $c = 1,58889$ *(siehe Seiden (2001)).* △

Der bestmögliche Kompetitivitätsfaktor ist derzeit nicht bekannt, das „Unsicherheitsintervall“ $1,5401 \leq c \leq 1,58889$ ist jedoch bereits sehr klein.

Ein noch erstaunlicheres Resultat gibt es für eine Variante des Online-Bin-Packing Problems. Wir nennen es *Online-Bin-Packing mit beschränktem Umpacken.* Hierbei darf der Online-Algorithmus jeweils k Behälter ($k \geq 2$ vorgegeben und fest) offen halten und zwischen den Behältern umpacken. Wenn der Algorithmus einen weiteren Behälter benötigt, muss er einen der gegenwärtigen k Behälter für immer schließen. Ein Vorteil besteht also in der Möglichkeit des Umpackens zwischen den k offenen Behältern, dafür dürfen aber einmal geschlossene Behälter nie wieder angefasst werden. Für dieses Problem ist der bestmögliche Kompetititvitätsfaktor bekannt.

(8.46) Satz.

(a) *Kein Online-Algorithmus für das Bin-Packing-Problem mit beschränktem Umpacken (k Kisten offen, k fest) hat einen Kompetitivitätsfaktor, der kleiner als* $1,69103$ *ist (siehe Lee und Lee (1985)).*

(b) *Es gibt einen Online-Algorithmus für das Bin-Packing-Problem mit beschränktem Umpacken, der bereits mit nur* 3 *offenen Kisten den Kompetitivitätsfaktor* $1,69103$ *garantiert (siehe Galambos und Woeginger (1993)).* △

Online-Probleme und -Algorithmen treten natürlich nicht nur in der kombinatorischen Optimierung auf. Das Buch Grötschel et al. (2001) enthält 37 Kapitel, die sich mit Online- und Echtzeit-Fragestellungen in allen Bereichen der Optimierung beschäftigen.

8.6 Duale Heuristiken, Relaxierungen

In den Abschnitten 8.1 – 8.5 haben wir uns damit beschäftigt, wie man (gute) zulässige Lösungen für ein kombinatorisches Optimierungsproblem findet. Mit den dort vorgestellten Methoden berechnet man obere Schranken (bzw. untere Schranken) für Minimierungsprobleme (bzw. Maximierungsprobleme). Nun wollen wir uns Sicherheit verschaffen, und zwar dadurch, dass wir Methoden angeben, die die Zielfunktionswerte von der „anderen Seite“ beschränken.

Dieses Thema ist meiner Meinung nach genau so wichtig, wie der Entwurf von Primalheuristiken. Denn nur durch die gleichzeitige Kenntnis guter unterer und oberer Schranken kann man sich ein Bild von der Qualität von Lösungen machen. Leider wird der Berechnung derartiger Schranken in der Literatur bisher wenig Beachtung geschenkt, in der Praxis scheint diese Möglichkeit weithin unbekannt zu sein.

Wie bereits gesagt, wollen wir Verfahren, die für ein Maximierungsproblem eine obere Schranke bzw. für ein Minimierungsproblem eine untere Schranke für den optimalen Zielfunktionswert bestimmen, *duale Heuristiken* nennen. Duale Heuristiken liefern i. A.

keine zulässigen Lösungen, ermöglichen aber – zusammen mit primalen Heuristiken – häufig gute Abschätzungen des optimalen Zielfunktionswertes.

Wir haben bereits mehrere duale Heuristiken kennengelernt, ohne dies explizit zu sagen. Denn bei jeder primalen Heuristik, für die wir eine Gütegarantie bewiesen haben, muss ten wir eine Abschätzung des Optimalwertes finden. Diese Abschätzung kann man als duale Heuristik ansehen. Wir lassen einige dieser Methoden direkt Revue passieren.

Beginnen wir mit der LIST-Heuristik (8.24) für das Parallel-Shop-Problem (8.22). Hier stehen m identische Maschinen parallel zur Verfügung, und n Aufgaben mit Bearbeitungsdauern $t_1, \ldots, t_n$ sind gegeben. Offenbar muss insgesamt $\sum_{i=1}^{n} t_i$ Bearbeitunszeit zur Verfügung gestellt werden. Verteilen wir diese gleichmäßig auf alle m Maschinen, so gilt für die optimale Lösung c_{opt}:

$$c_{\text{opt}} \geq \frac{1}{m} \sum_{i=1}^{n} t_i.$$

Diese triviale Abschätzung haben wir in (8.6) benutzt. Wir können sie natürlich als duale Heuristik auffassen.

Für die Christofides-Heuristik (8.5)(f) haben wir in Satz (8.8) gezeigt, dass sie ein $\frac{1}{2}$-approximatives Verfahren für das symmetrische TSP mit Dreiecksungleichung ist. Ist T_{CH} eine durch die Christofides-Heuristik gefundene Tour, so gilt daher für jede optimale Tour T_{opt} die Abschätzung $c(T_{\text{opt}}) \geq \frac{2}{3} c(T_{\text{CH}})$. Also kann man aus der (primalen) Christofides-Heuristik eine duale Heuristik für das euklidische symmetrische TSP machen.

Zwei Relaxierungstechniken

Derzeit werden im wesentlichen zwei grundsätzliche Relaxierungstechniken für kombinatorische Optimierungsprobleme benutzt. Diese wollen wir hier kurz beschreiben und an Beispielen erläutern. Eine Methode, Relaxierungen zu verbessern, wird im nächsten Abschnitt vorgeführt.

Generell ist eine Relaxierung eine „Einbettung“ eines Problems in ein größeres. Uns interessieren die beiden folgenden Relaxierungen.

(8.47) Definition. *Gegeben sei ein kombinatorisches Optimierungsproblem* $(E, \mathcal{I}, c)$.

(a) *Jedes kombinatorische Optimierungsproblem* $(E', \mathcal{I}', c')$, *in das* $(E, \mathcal{I}, c)$ *eingebettet werden kann, nennen wir eine* kombinatorische Relaxierung *von* $(E, \mathcal{I}, c)$. *„Einbettung“ heißt hier, es gibt eine injektive Abbildung* $\varphi : E \to E'$ *mit* $\varphi(I) \in \mathcal{I}'$ *für alle* $I \in \mathcal{I}$ *und* $c(I) = c'(\varphi(I))$ *für alle* $I \in \mathcal{I}$.

(b) *Jedes lineare Programm* max *(oder* min*)* $c^T x$, $Ax \leq b$, $x \geq 0$ *mit der Eigenschaft* $\{x \in \mathbb{K}^E \mid Ax \leq b, x \geq 0, x \text{ ganzzahlig}\}$ *ist die Menge der Inzidenzvektoren von* $\mathcal{I}$ *heißt* LP-Relaxierung *von* $(E, \mathcal{I}, c)$. △

Was können wir nun mit Relaxierungen anfangen? Angenommen $(E, \mathcal{I}, c)$ ist ein kombinatorisches Maximierungsproblem. Aus der Einbettungsvorschrift folgt direkt

$$\max\{c'(I') \mid I' \in \mathcal{I}'\} \geq \max\{c(I) \mid I \in \mathcal{I}\} \tag{8.48}$$

für jede kombinatorische Relaxierung von $(E, \mathcal{I}, c)$ und analog

$$\max\{c^T x \mid Ax \leq b, x \geq 0\} \geq \max\{c(I) \mid I \in \mathcal{I}\} \tag{8.49}$$

für jede LP-Relaxierung von $(E, \mathcal{I}, c)$. Entsprechendes gilt für Minimierungsprobleme. Daraus ergibt sich, daß jeder Algorithmus zur Lösung einer kombinatorischen oder LP-Relaxierung eines kombinatorischen Optimierungsproblems als Dualheuristik angesehen werden kann. Der Erfolg derartiger Dualheuristiken hängt natürlich ganz entscheidend von der Wahl der Relaxierung ab. Bei kombinatorischen Relaxierung ist das Erreichen von zwei Zielen wünschenswert:

(a) $|\mathcal{I}'| - |\mathcal{I}|$ soll möglichst klein sein, d. h. $\mathcal{I}'$ soll nur einige wenige Elemente mehr als $\mathcal{I}$ enthalten. Dadurch besteht die begründete Hoffnung, daß die Optimallösung über $\mathcal{I}'$ bereits in $\mathcal{I}$ liegt bzw. die Optimalwerte der beiden Probleme nicht stark voneinander abweichen.

(b) $(E', \mathcal{I}', c')$ ist effizient (polynomial) lösbar.

Bei LP-Relaxierungen sollten das Polyeder $\{x \mid Ax \leq b, x \geq 0\}$ möglichst „nahe“ an $\text{conv}\{\chi^I \mid I \in \mathcal{I}\}$ liegen, und ferner sollte das LP $\max c^T x$, $Ax \leq b$, $x \geq 0$ möglichst effizient lösbar sein.

Das Thema LP-Relaxierungen und Lösung derartiger linearer Programme haben wir in den Kapiteln 3 und 7 behandelt und sind dort genauer auf die derzeit gegebenen Möglichkeiten eingegangen, kombinatorische Optimierungsprobleme mit Hilfe linearer Optimierungsverfahren zu lösen.

Wir geben nun einige weitere Beispiele für kombinatorische und LP-Relaxierungen an.

(8.50) Bemerkung (2-Matching Relaxierung des symmetrischen TSP). Ein perfektes 2-Matching (kurz 2-*Matching*) ist eine Kantenmenge M eines Graphen G, so dass jeder Knoten auf genau zwei Kanten liegt. 2-Matchings sind also Vereinigungen von knotendisjunkten Kreisen, so dass jeder Knoten auf einem Kreis liegt. Offenbar ist jede Tour ein 2-Matching. Daraus folgt für jede Zielfunktion $c : \mathbb{K}^E \to \mathbb{K}$

$$\min\{c(M) \mid M \text{ 2-Matching}\} \leq \min\{c(T) \mid T \text{ Tour}\}.$$

Minimale 2-Matchings kann man mit einem Algorithmus von Edmonds in polynomialer Zeit bestimmen. Jedoch sind bisher noch keine wirklich effizienten 2-Matching Codes entwickelt worden, so dass die Nützlichkeit der 2-Matching-Relaxierung des symmetrischen Travelling-Salesman-Problems noch nicht intensiv untersucht wurde. △

Um ein Beispiel für die Abweichung des Wertes einer 2-Matching Lösung von der minimalen Tourlänge zu geben, habe ich mit LP-Techniken (Schnittebenenverfahren, siehe Kapitel 7) ein minimales 2-Matching für ein 120-Städte Deutschland-Problem berechnet. Dieses minimale 2-Matching ist in Abbildung 8.15 zu sehen. Die Länge dieses 2-Matchings beträgt 6 694 km, während die optimale Tourlänge 6 942 km beträgt. Das minimale 2-Matching ist also 3,57% kürzer als die Optimaltour.

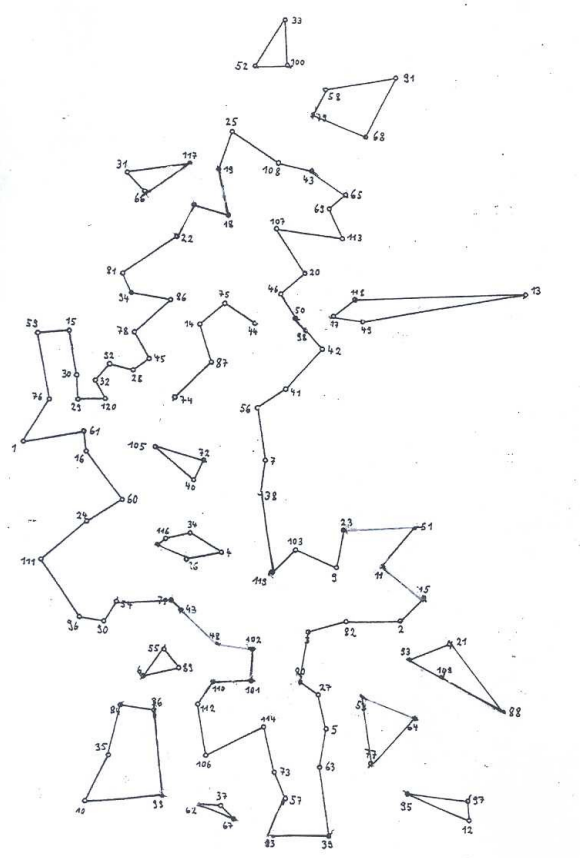


Abbildung 8.15: 2-Matching für das 120-Städte Deutschland-Problem, Länge 6 694 km

(8.51) Bemerkung (1-Baum-Relaxierung des symmetrischen TSP). Sei $G = (V, E)$ ein zweifach zusammenhängender Graph und v ein Knoten von G. Jede Kantenmenge B, die aus einem aufspannenden Baum von $G - v$ und zwei Kanten, die mit v inzidieren, besteht, heißt 1-*Baum (von G)*. Die Menge aller 1-Bäume von G ist das Basissystem eines Matroids auf E, siehe Kapitel 2. Man kann also einen minimalen 1-Baum von G mit dem Greedy-Algorithmus (2.18) bestimmen. Konkret berechnet man zunächst mit einem der Algorithmen aus ADM Ieinen minimalen aufspannenden Baum von $G - v$, dazu fügt man zwei Kanten, die mit v inzidieren, deren Gewichtsumme minimal ist.

Jede Tour des K_n ist ein 1-Baum des K_n, denn sie besteht aus einem hamiltonschen Weg von $K_n - v$ und den beiden Kanten, die v mit den Endknoten des hamiltonschen Weges verbinden. Daher gilt für jede Zielfunktion $c \in \mathbb{K}^E$:

$$\min\{c(B) \mid B\text{1-Baum}\} \leq \min\{c(T) \mid T \text{ Tour}\}. \qquad \triangle$$

Wie wir oben angemerkt haben, sind 1-Bäume sehr einfach zu berechnen. Für das 120-Städte Deutschland-Problem ist ein minimaler 1-Baum (v = Knoten 13 = Berlin) in Abbildung 8.16 dargestellt. Seine Länge beträgt 6 025 km. Er ist also 13,21% kürzer als

die Optimaltour. I. A. ist die 1-Baum-Relaxierung erheblich schlechter als die 2-Matching-Relaxierung, aber sie ist viel einfacher zu berechnen. In folgenden Abschnitt werden wir sehen, wie man die 1-Baum-Relaxierung erheblich verbessern kann.

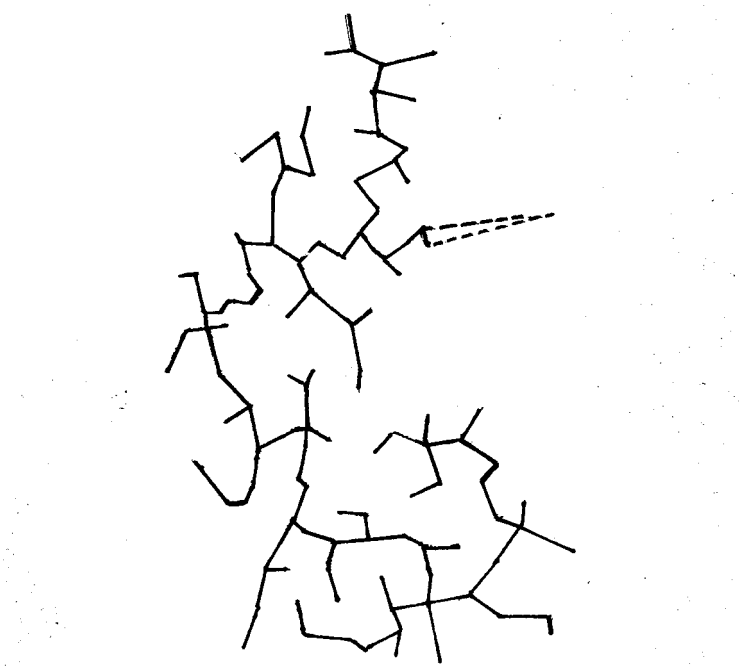


Abbildung 8.16: Minimaler 1-Baum, Länge 6 025 km

(8.52) Bemerkung (Zuordnungsrelaxierung des asymmetrischen TSP). Gegeben sei ein vollständiger Digraph (ohne Schleifen) $D = (V, A)$ mit Bogengewichten $c_a \in \mathbb{K}$ für alle $a \in A$. Gesucht ist eine Bogenmenge Z minimalen Gewichts, so dass jeder Knoten jeweils Anfangsknoten und Endknoten genau eines Bogens aus Z ist. Jede Zuordnung ist also die Vereinigung knotendisjunkter gerichteter Kreise, so dass jeder Knoten auf genau einem Kreis liegt. Offenbar ist jede Tour eine Zuordnung. △

Diese Version des Zuordnungsproblems ist ein Spezialfall des Problems, einen kostenminimalen Fluss in einem Digraphen zu finden. Für dieses Zuordnungsproblem gibt es sehr effiziente Algorithmen (Worst-case-Laufzeit $O(n^3)$, praktisch schneller). Es ist die „beliebteste“ Relaxierung des asymmetrischen Travelling-Salesman-Problems und wird mit gutem Erfolg in Branch & Bound-Verfahren für das asymmetrische Travelling-Salesman-Problems eingesetzt, siehe Kapitel 6.

Die Zuordnungsrelaxierung des asymmetrischen Travelling-Salesman-Problems nimmt eine Zwitterstellung zwischen den beiden Relaxierungstypen ein. Wie man leicht sieht ist das folgende ganzzahlige Programm eine 0/1-Formulierung des asymmetrischen Travelling-

Salesman-Problems

$$
\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{n}\sum_{j=1,i\neq j}^{n} c_{ij}x_{ij} & \\
\text{(i)} & \displaystyle\sum_{i=1,i\neq j}^{n} x_{ij} = 1 & \text{für alle } j \in V \\
\text{(ii)} & \displaystyle\sum_{j=1,j\neq i}^{n} x_{ij} = 1 & \text{für alle } i \in V \\
\text{(iii)} & \displaystyle\sum_{i\in W}\sum_{j\in W,i\neq j} x_{ij} \leq |W| - 1 & \text{für alle } W \subseteq V, 3 \leq |W| \leq n-1 \\
\text{(iv)} & x_{ij} \geq 0 & \text{für alle } (i,j) \in A \\
\text{(v)} & x_{ij} \text{ ganzzahlig} & \text{für alle } (i,j) \in A
\end{array}
\tag{8.53}
$$

Die Ungleichungen (iii) sind die Kurzzyklus-Bedingungen. Lässt man diese aus (8.53) weg, so erhält man eine Formulierung des Zuordnungsproblems aus (8.52) als ganzzahliges Programm. Aus den Resultaten von ADM I, Kapitel 13 folgt, dass in diesem Programm die Ganzzahligkeitsbedingung überflüssig ist, da alle Basislösungen des Systems (i), (ii), (iv) ganzzahlig sind. Das Zuordnungsproblem ist also auch eine gewisse LP-Relaxierung des asymmetrischen Travelling-Salesman-Problems.

Damit wollen wir kurz auf weitere LP-Relaxierungen eingehen.

(8.54) Bemerkung (LP-Relaxierungen des Stabile-Mengen-Problems (ADM I, (3.13))). Sei $G = (V, E)$ ein Graph mit Knotengewichten c_v für alle $v \in V$. Gesucht ist eine stabile Menge maximalen Gewichts. Da keine stabile Knotenmenge $S \subseteq V$ eine Kante uv enthalten kann, sieht man sofort, dass das LP

$$
\begin{array}{lll}
\max & \displaystyle\sum_{v\in V} c_v x_v & \\
\text{(i)} & x_u + x_v \leq 1 & \text{für alle } uv \in E \\
\text{(ii)} & 0 \leq x_v \leq 1 & \text{für alle } v \in V \\
\text{(iii)} & x_v \text{ ganzzahlig} & \text{für alle } v \in V
\end{array}
\tag{8.55}
$$

eine 0/1-Formulierung des Stabile-Mengen-Probems ist. Das LP, das aus (8.55) durch Weglassen der Ganzzahligkeitsbedingung (iii) entsteht, ist somit eine LP-Relaxierung des Stabile-Mengen-Problems. Man kann beweisen, dass alle Basislösungen dieses LPs genau dann ganzzahlig sind, wenn G ein bipartiter Graph ist. Diese LP-Relaxierung ist einfach (mit Spezialalgorithmen) lösbar, liefert allerdings i. A. keine allzu guten oberen Schranken für das Maximalgewicht einer stabilen Menge. △

Man kann diese LP-Formulierung (8.55) auf vielfältige Weise verschärfen. Umfangreiche Untersuchungen dieser Art sind in Grötschel et al. (1993) zu finden.

(8.56) Bemerkung (LP-Relaxierung des Knotenfärbungsproblems (ADM I, (3.14))). Gegeben sei ein Graph $G = (V, E)$ mit Knotengewichten $b_v \in \mathbb{Z}_+$, $v \in V$. Gesucht ist eine Folge stabiler Mengen (eine *Färbung*) $S_1, \ldots, S_t$, so dass jeder Knoten in mindestens b_v dieser Mengen liegt und t minimal ist. Jeder stabilen Menge $S \subseteq V$ ordnen wir eine ganzzahlige Variable λ_S zu. (Die Anzahl der stabilen Mengen eines Graphen ist i. A. exponentiell in $|V|$!) Die Variable λ_S gibt an, wie oft die stabile Menge S in einer Färbung $S_1, \ldots, S_t$ vorkommt. Ist $\mathcal{S}$ die Menge der stabilen Teilmengen von V, so ist folglich

$$\begin{aligned} &\min \quad \sum_{S\in\mathcal{S}} \lambda_S \\ \text{(i)} \quad &\sum_{S\in\mathcal{S}, v\in S} \lambda_S \geq b_v && \text{für alle } v \in V \\ \text{(ii)} \quad &\lambda_S \geq 0 && \text{für alle } S \in \mathcal{S} \\ \text{(iii)} \quad &\lambda_S \text{ ganzzahlig} && \text{für alle } S \in \mathcal{S} \end{aligned} \tag{8.57}$$

eine Formulierung des Knotenfärbungsproblems als ganzzahliges lineares Programm (in exponentiell vielen Variablen). Lässt man die Ganzzahligkeitsbedingung (iii) in (8.57) weg, so erhält man eine LP-Relaxierung des Färbungsproblems.

Es ist natürlich unklar, wie man ein derartig großes LP lösen kann. Wir wollen daher auf die LP-Dualität zurückgreifen und weiter relaxieren. Das zu diesem LP duale lineare Programm lautet

$$\begin{aligned} &\max \quad \sum_{v\in V} b_v x_v \\ \text{(i)} \quad &\sum_{v\in S} x_v \leq 1 && \text{für alle } S \subseteq \mathcal{S} \\ \text{(ii)} \quad &x_v \geq 0 && \text{für alle } v \in V \end{aligned} \tag{8.58}$$

Das Programm (8.58) hat $|V|$ Variablen und $|\mathcal{S}| + |V|$, also i. A. exponentiell viele Nebenbedingungen. Aus der Dualitätstheorie wissen wir, dass jede Lösung von (8.58) einen Wert hat, der nicht größer ist als der Wert jeder Lösung der LP-Relaxierung von (8.57), also insbesondere liefert jede zulässige Lösung von (8.58) eine untere Schranke von (8.57) und somit eine untere Schranke für den Wert einer Knotenfärbung.

Man kann zeigen, dass Probleme des Typs (8.58) $\mathcal{NP}$-schwer sind, aber da jede Lösung von (8.58) eine untere Schranke für (8.57) liefert, kann man versuchen mit Heuristiken „gute“ Lösungen von (8.58) zu finden.

Offenbar ist jeder Inzidenzvektor einer Clique von G eine (ganzzahlige) Lösung von (8.58). Also liefert jede Heuristik für das Cliquenproblem (siehe ADM I, (3.13)) eine untere Schranke für (8.57). △

Lagrange-Relaxierungen

Wir wollen nun eine Methode vorstellen, mit der man gegebene Relaxierungen verbessern kann. Wir werden die Idee zunächst an der 1-Baum-Relaxierung des symmetrischen TSPs

erklären (hierbei ist sie auch von Held & Karp (1970) entwickelt worden), und sie dann in voller Allgemeinheit formulieren. Wir beginnen mit einer simplen Beobachtung.

(8.59) Satz. *Gegeben sei ein symmetrisches TSP mit Entfernungen $c_{ij}, 1 \le i < j \le n$. Seien λ_i, $i = 1, \ldots, n$, Knotengewichte, $\lambda := \sum_{i=1}^{n} \lambda_i$ und*

$$c'_{ij} := c_{ij} + \lambda_i + \lambda_j, \quad 1 \le i < j \le n.$$

Für das modifizierte TSP mit Entfernungen c'_{ij} gilt:

$$c'(T) = c(T) + 2\sum_{i=1}^{n} \lambda_i = c(T) + 2\lambda \qquad \text{für alle Touren } T,$$

$$c'(B) = c(B) + \sum_{i=1}^{n} d_B(i)\lambda_i \qquad \text{für alle 1-Bäume } B,$$

wobei $d_B(i)$ den Grad des Knotens i im 1-Baum B bezeichnet. Das heißt, die optimalen Touren bezüglich c bleiben auch optimal bezüglich c', während ein optimaler 1-Baum bezüglich c nicht unbedingt optimal bezüglich c' ist. △

Beweis. Offensichtlich. □

(8.60) Korollar. *Gegeben sei ein symmetrisches TSP mit Entfernungen c_{ij}, $1 \le i < j \le n$. Dann ist*

$$f^* := \max_{\lambda \in \mathbb{R}^n} \Big(\min_{B \text{ 1-Baum}} \big(c(B) + \sum_{i=1}^{n} (d_B(i) - 2)\lambda_i \big) \Big)$$

eine untere Schranke für die Länge der optimalen Tour. △

Beweis. Jede Tour ist ein 1-Baum, und es ist $d_T(i) = 2$ für jeden Knoten i in jeder Tour T. Also gilt für jede Tour T

$$c(T) + \sum_{i=1}^{n} (d_T(i) - 2)\lambda_i = c(T),$$

d. h. für jedes $\lambda \in \mathbb{R}^n$ ist der Wert des minimalen 1-Baumes höchstens so groß wie die Länge der kürzesten Tour. □

(8.61) Beispiel. Wir betrachten erneut das 6-Städte TSP aus Beispiel (8.6). Die Entfernungen sind in Tabelle 8.1 angegeben. Wir eliminieren Wuppertal und erhalten den in Abbildung 8.17 dargestellten minimalen 1-Baum mit Länge 403 km. Wir führen als Knotengewichte

$$\lambda_K = 15\,, \quad \lambda_F = -120\,, \quad \lambda_A = -5\,, \quad \lambda_D = 0\,, \quad \lambda_B = 0$$

ein und erhalten die in Tabelle 8.3 wiedergegebene Entfernungsmatrix. Der bezüglich der neuen Entferungen kürzeste 1-Baum hat die Länge 443 km und ist in Abbildung 8.18 dargestellt. Damit haben wir also eine bessere untere Schranke für die optimale Tour gefunden. △

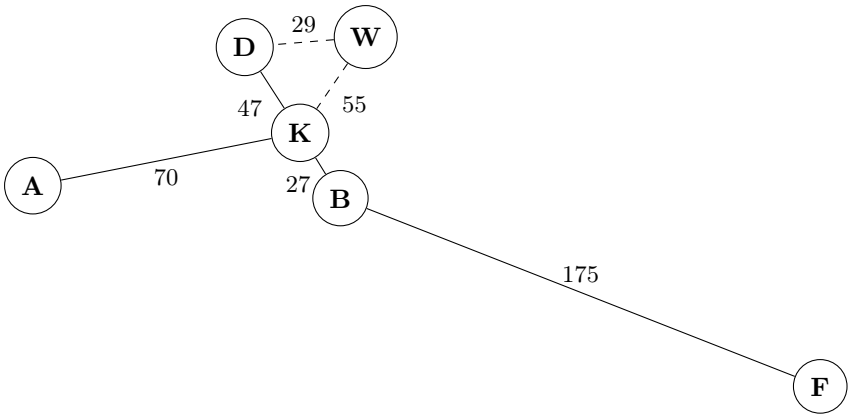


Abbildung 8.17: Minimaler 1-Baum bezüglich Wuppertal

		-5	0	0	-120	15	0
		A	B	D	F	K	W
-5	A	–	86	75	134	80	116
0	B	86	–	77	55	42	84
0	D	75	77	–	112	62	29
-120	F	134	55	112	–	84	116
15	K	80	42	62	84	–	70
0	W	116	84	29	116	70	–

Tabelle 8.3: Knotengewichte und modifizierte Entfernungen für 6 Städte im Rheinland

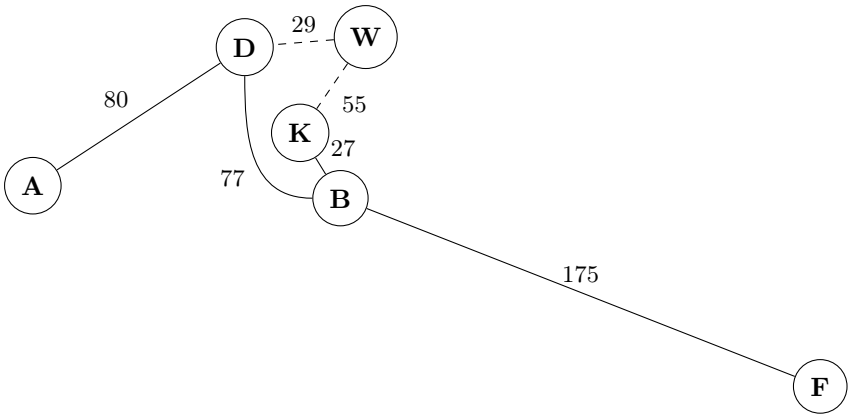


Abbildung 8.18: Minimaler 1-Baum bezüglich der Knotengewichte in Tabelle 8.3

Wir wissen zwar, wie wir 1-Bäume schnell berechnen können, aber wie soll das in (8.60) formulierte freie Maximierungsproblem (über alle $\lambda \in \mathbb{R}^n$) gelöst werden? Wie finden wir f^*?

Auf folgende Weise können wir eine Funktion $f : \mathbb{R}^n \to \mathbb{R}$ definieren:

$$f(\lambda) := \min_{B \text{ 1-Baum}} \Big(c(B) + \sum_{i=1}^{n} (d_B(i) - 2)\lambda_i \Big) \tag{8.62}$$

d. h. f ist eine Funktion, deren Auswertung die Berechnung der Optimallösung eines Optimierungsproblems erfordert. Das Problem, f^* zu finden, ist damit das unbeschränkte Optimierungsproblem

$$\max_{\lambda \in \mathbb{R}^n} f(\lambda).$$

Um diese Situation zu analysieren, verallgemeinern wir die vorliegende Fragestellung: Wir betrachten ein ($\mathcal{NP}$-schwieriges) ganzzahliges lineares Programm der folgenden Form

$$\begin{aligned} c^* := \min\ & c^T x \\ & Ax = b \\ & Dx \leq d \\ & x \geq 0 \\ & x \text{ ganzzahlig}, \end{aligned} \tag{8.63}$$

wobei A eine (m, n)-Matrix sei. Wir setzen

$$\begin{aligned} P &:= \{x \in \mathbb{R}^n \mid Ax = b, Dx \leq d, x \geq 0, x \text{ ganzzahlig}\}, \\ Q &:= \{x \in \mathbb{R}^n \mid Dx \leq d, x \geq 0, x \text{ ganzzahlig}\}. \end{aligned}$$

(8.64) Definition. *Die Funktion $f : \mathbb{R}^m \to \mathbb{R}$ definiert durch*

$$f(\lambda) := \min\{c^T x + \lambda^T (Ax - b) \mid x \in Q\}$$

heißt Lagrange-Funktion *von* (8.63)*, der Vektor $\lambda \in \mathbb{R}^m$ heißt Vektor der* Lagrange-Multiplikatoren*. Das Optimierungsproblem*

$$\max_{\lambda \in \mathbb{R}^m} f(\lambda) \tag{8.65}$$

heißt Lagrange-Relaxierung *von* (8.63). △

Die in (8.62) definierte Funktion ist die Lagrange-Funktion des symmetrischen TSP bezüglich der 1-Baum Relaxierung. Denn, wie wir wissen, ist das folgende Programm

eine ganzzahlige Formulierung des TSP:

$$
\begin{array}{lrll}
 & \min \quad c^T x & & \\
\text{(i)} & x(\delta(v)) = 2 & 1 \le v \le n & \\
\text{(ii)} & x(E(W)) \le |W| - 1 & \forall\, W \subseteq V \setminus \{1\}, |W| \ge 3 & \\
\text{(iii)} & x(E(V \setminus \{1\})) \ge n - 2 & & \\
\text{(iv)} & x(\delta(1)) \le 2 & & \\
\text{(v)} & x(\delta(1)) \ge 2 & & \\
\text{(vi)} & x \le 1 & & \\
\text{(vii)} & x \ge 0 & & \\
\text{(viii)} & x \text{ ganzzahlig.} & &
\end{array}
\tag{8.66}
$$

Schreiben wir das Gleichungssystem (i) von (8.66) als $Ax = b$ und die Ungleichungen (ii)–(vi) als $Dx \le d$, so ist die in (8.62) definierte Funktion genau die Lagrange-Funktion des TSP-IPs.

(8.67) Satz. *Seien Q und P (wie oben definiert) nicht leer und Q endlich. Dann gilt:*

(a) *Die Lagrange-Funktion $f(\lambda) = \min_{x \in Q}(c^T x + \lambda^T (Ax - b))$ ist konkav, stückweise linear und nach oben beschränkt.*

(b) *Sei $\lambda^* \in \mathbb{R}^m$ mit*

$$f(\lambda^*) = \max_{\lambda \in \mathbb{R}^m} f(\lambda)$$

eine Optimallösung der Lagrange-Relaxierung von (8.63)*, so gilt*

$$f(\lambda^*) \le c^*,$$

d. h. $f(\lambda^*)$ ist eine untere Schranke für den Optimalwert von (8.63)*.* △

Beweis. (a) Wir können für jeden Vektor $x \in Q$ eine affine Funktion g_x in λ wie folgt definieren:

$$g_x(\lambda) := c^T x + \lambda^T (Ax - b).$$

Da Q endlich ist, ist f als Minimum endlich vieler affiner Funktionen darstellbar, d. h.

$$f(\lambda) = \min_{x \in Q} g_x(\lambda).$$

Wir zeigen

- **f ist konkav.**
  Es ist zu beweisen: $f(\alpha\lambda + (1-\alpha)\mu) \ge \alpha f(\lambda) + (1-\alpha) f(\mu)$ für alle $0 \le \alpha \le 1$ und

für alle $\lambda, \mu \in \mathbb{R}^m$.

$$\begin{aligned} f(\alpha\lambda + (1-\alpha)\mu) &= \min_{x \in Q} g_x(\alpha\lambda + (1-\alpha)\mu) \\ &= \min_{x \in Q} (\alpha g_x(\lambda) + (1-\alpha) g_x(\mu)) \\ &\geq \alpha \min_{x \in Q} g_x(\lambda) + (1-\alpha) \min_{x \in Q} g_x(\mu) \\ &= \alpha f(\lambda) + (1-\alpha) f(\mu). \end{aligned}$$

- **f ist stückweise linear.**
  Sei für jedes $x \in Q$: $L_x := \{\lambda \in \mathbb{R}^m \mid f(\lambda) = g_x(\lambda)\}$. Es gilt natürlich $\bigcup_{x \in Q} L_x = \mathbb{R}^m$. Wenn wir zeigen können, dass L_x konvex ist, dann gilt $f(\lambda) = g_x(\lambda)\ \forall\, \lambda \in L_x$. Somit ist f linear auf L_x, d. h. stückweise linear auf $\mathbb{R}^m$.
  L_x konvex $\iff$ $\alpha\lambda + (1-\alpha)\mu \in L_x$ für alle $0 \leq \alpha \leq 1$ und für alle $\lambda, \mu \in L_x$.

  $$\begin{aligned} g_x(\alpha\lambda + (1-\alpha)\mu) \ &\geq\ f(\alpha\lambda + (1-\alpha)\mu) \ \geq\ \alpha f(\lambda) + (1-\alpha) f(\mu) \\ &=\ \alpha g_x(\lambda) + (1-\alpha) g_x(\mu) \ =\ g_x(\alpha\lambda + (1-\alpha)\mu). \end{aligned}$$

  Daraus folgt: L_x ist konvex für alle $x \in Q$, und somit ist f stückweise linear.

- **f ist nach oben beschränkt.**
  Sei $\overline{x} \in P$ beliebig. Dann gilt für jeden beliebigen Vektor von Lagrange-Multiplikatoren $\lambda \in \mathbb{R}^m$:

  $$f(\lambda) = \min_{x \in Q} g_x(\lambda) \leq g_{\overline{x}}(\lambda) = c^T \overline{x}.$$

  Daraus folgt insbesondere, dass $f(\lambda) \leq c^*$ für alle $\lambda \in \mathbb{R}^m$ gilt.

(b) ist offensichtlich. □

Es besteht nun das Problem, das Maximum der Lagrange-Funktion f zu bestimmen. Die Standardverfahren der nichtlinearen Optimierung kann man hier nicht anwenden, da die Funktion f i. A. nicht differenzierbar ist. Eine Verallgemeinerung des Gradientenkonzepts aus der Analysis führt jedoch zu einer Lösungsmöglichkeit.

(8.68) Definition. *Sei $f : \mathbb{R}^m \to \mathbb{R}$ eine konkave Funktion, dann heißt ein Vektor $u \in \mathbb{R}^m$ ein* Subgradient *an f im Punkte $\lambda_0 \in \mathbb{R}^m$, falls gilt*

$$f(\lambda) - f(\lambda_0) \leq u^T(\lambda - \lambda_0) \quad \textit{für alle } \lambda \in \mathbb{R}^m.$$

Die Menge

$$\partial f(\lambda_0) := \{u \in \mathbb{R}^m \mid u \textit{ ist Subgradient an } f \textit{ in } \lambda_0\}$$

heißt Subdifferential *von f in λ_0.* △

(8.69) Bemerkung. Ist die konkave Funktion f differenzierbar in λ_0, so gilt

$$\partial f(\lambda_0) = \{\bigtriangledown f(\lambda_0)\},$$

wobei $\bigtriangledown f(\lambda_0)$ den Gradienten von f an der Stelle λ_0 bezeichnet. △

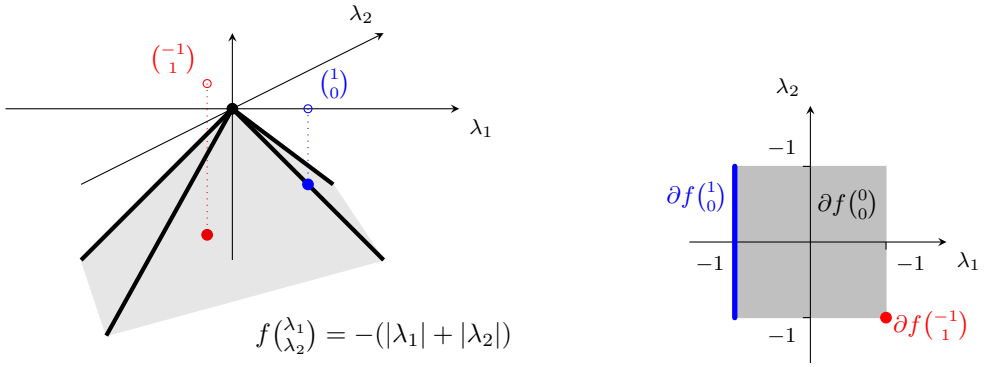


Abbildung 8.19: Graph einer konkaven Funktion f und einige Subdifferentiale

Der folgende Satz verallgemeinert die uns aus der Schule bekannte Tatsache, dass ein Punkt eine differenzierbare konkave Funktion f maximiert, wenn die Ableitung von f in diesem Punkt Null ist.

(8.70) Satz. *Ein Vektor $\lambda^* \in \mathbb{R}^m$ maximiert die konkave Funktion f genau dann, wenn $0 \in \partial f(\lambda^*)$.* △

Beweis. Falls $0 \in \partial f(\lambda^*)$, dann ergibt die Subgradientenungleichung der Definition:

$$f(\lambda) - f(\lambda^*) \leq 0^T(\lambda - \lambda^*) \implies f(\lambda) \leq f(\lambda^*) \text{ für alle } \lambda \in \mathbb{R}^m.$$

Falls $f(\lambda^*)$ maximal ist, so ist $f(\lambda) \leq f(\lambda^*)$ für alle $\lambda \in \mathbb{R}^m$, also $f(\lambda) - f(\lambda^*) \leq 0 = 0^T(\lambda - \lambda^*)$. Daraus folgt $0 \in \partial f(\lambda^*)$. □

(8.71) Beispiel. Für die konkave, stückweise lineare Funktion

$$f : \mathbb{R}^2 \to \mathbb{R}, (\lambda_1, \lambda_2) \mapsto -(|\lambda_1| + |\lambda_2|)$$

ist das Subdifferential gegeben durch

$$\partial f(\lambda_0) = \begin{cases} [-1,1] \times [-1,1] & \text{falls } \lambda_0 = (0,0)^T, \\ \{(-\operatorname{sign}\alpha, \gamma)^T \mid -1 \leq \gamma \leq 1\} & \text{falls } \lambda_0 = (\alpha, 0)^T, \alpha \in \mathbb{R} \setminus \{0\}, \\ \{(\gamma, -\operatorname{sign}\alpha)^T \mid -1 \leq \gamma \leq 1\} & \text{falls } \lambda_0 = (0, \alpha)^T, \alpha \in \mathbb{R} \setminus \{0\}, \\ \{(-\operatorname{sign}\alpha, -\operatorname{sign}\beta)^T\} & \text{falls } \lambda_0 = (\alpha, \beta)^T, \alpha, \beta \in \mathbb{R} \setminus \{0\}, \end{cases}$$ △

siehe Abbildung 8.19.

Für die speziellen Lagrange-Funktionen, die wir hier betrachten, können wir die Subdifferentiale einfach berechnen.

(8.72) Satz. *Sei Q nicht leer und endlich und $f(\lambda) := \min_{x \in Q}(c^T x + \lambda^T(Ax - b))$, so gilt folgendes: Setzen wir für $\lambda_0 \in \mathbb{R}^m$, $L_0 := \{x_0 \in \mathbb{R}^n \mid f(\lambda_0) = c^T x_0 + \lambda_0^T(Ax_0 - b)\}$, so ist*

$$\partial f(\lambda_0) = \operatorname{conv}\{Ax_0 - b \mid x_0 \in L_0\}.$$ △

Aus Zeitgründen können wir auf den Beweis dieses Satzes hier nicht eingehen. Der Satz hat einen einfachen geometrischen Gehalt. Für jedes $\lambda_0 \in \mathbb{R}^m$ ist $\partial f(\lambda_0)$ ein Polytop. Dieses Polytop erhält man wie folgt. Man nehme alle Punkte $x_0 \in \mathbb{R}^n$, die das Minimum des Optimierungsproblems $\min\{c^T x + \lambda_0^T(Ax - b) \mid x \in Q\}$ realisieren, also alle Optimalwerte bezüglich des Vektors der Lagrange-Multiplikatoren λ_0. Dann bilde man die konvexe Hülle all dieser Punkte und erhält das Subdifferential $\partial f(\lambda_0)$.

(8.73) Beispiel. Im Falle des symmetrischen TSPs gilt für einen Vektor von Knotengewichten $\lambda := (\lambda_1, \ldots, \lambda_n)$:

$$\partial f(\lambda) = \operatorname{conv}\Big\{ \begin{pmatrix} d_B(1) - 2 \\ d_B(2) - 2 \\ \vdots \\ d_B(n) - 2 \end{pmatrix} \mid B \text{ optimaler 1-Baum bezüglich } c' \Big\}.$$

Hat man einen optimalen 1-Baum B bezüglich λ, so kann man also einen Subgradienten der Lagrange-Funktion f direkt aus den Knotengraden von B ablesen. △

Zur Lösung der Optimierungsaufgabe (8.65) ist die folgende Methode vorgeschlagen worden:

(8.74) Algorithmus (Allgemeines Subgradientenverfahren).
Eingabe: Konkave Funktion $f : \mathbb{R}^m \to \mathbb{R}$
Ausgabe: $\max\{f(\lambda) \mid \lambda \in \mathbb{R}^m\}$

1. Wähle $\lambda_0 \in \mathbb{R}^m$ beliebig, setze $j := 0$.
2. Berechne $f(\lambda_j)$ und $\partial f(\lambda_j)$.
3. Ist ein (noch zu definierendes) STOP-Kriterium erfüllt?
   Falls ja, so ist eine „gute“ untere Schranke für das Maximum oder das Maximum selbst gefunden. $\longrightarrow$ STOP.
   Falls nein, gehe zu Schritt 4.
4. Wähle neues λ_{j+1}, setze z. B.

   $$\lambda_{j+1} := \lambda_j + t_j \frac{u_j}{\|u_j\|},$$

   wobei $t_j \in \mathbb{R}$ eine Schrittlänge ist und $u_j \in \partial f(\lambda_j)$.
5. Setze $j := j + 1$ und gehe zu Schritt 2. △

Bezüglich dieses Verfahrens kann man Folgendes beweisen.

(8.75) Satz (Polyak, 1967). *Falls die konkave Funktion $f : \mathbb{R}^m \to \mathbb{R}$ ein Maximum besitzt und falls die Folge $(t_j)_{j \in \mathbb{N}}$ der Schrittlängen die Bedingungen*

(a) $t_j > 0$ *für alle* $j \in \mathbb{N}$,

(b) $\lim_{j\to\infty} t_j = 0$,

(c) $\sum_{j=0}^{\infty} t_j = \infty$

erfüllt, so gilt

$$\lim_{j\to\infty} f(\lambda_j) = \max_{\lambda\in\mathbb{R}^m} f(\lambda).$$

△

Bei konkreten Rechnungen zeigt sich allerdings im allgemeinen, dass die (nach dem Satz von Polyak theoretisch existierende) Konvergenz praktisch nicht erreicht wird. Abbildung 8.20 zeigt einen typischen Verlauf des Wertes $f(\lambda_j)$, abhängig von der Schrittzahl j.

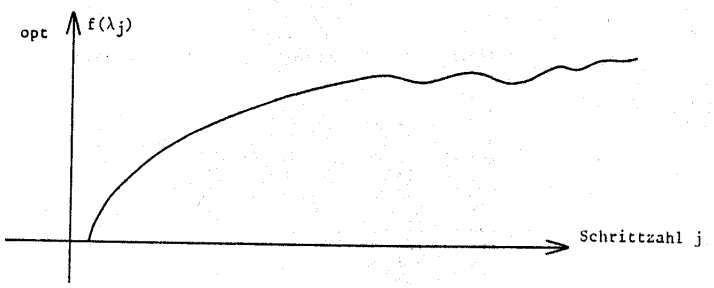


Abbildung 8.20: Typisches Konvergenzverhalten beim Subgradientenverfahren

Mit anderen Schrittlängenparametern kann man praktisch i. A. ein schnelleres Anwachsen der Funktionswerte $f(\lambda_j)$ erreichen, jedoch theoretisch nicht unbedingt eine konvergente Methode. Da es sich ja hier „nur“ um eine Heuristik zur Bestimmung unterer Schranken handelt, ist der Konvergenzaspekt gleichgültig. Man bricht einfach nach einer bestimmten Rechenzeit oder einem STOP-Kriterium ab und benutzt den bisher besten Zielfunktionswert als untere Schranke.

Aus empirischer Erfahrung kann man festhalten, dass die Schrittlängen t_j problemabhängig zu wählen sind und dass sie i. A. durch experimentelle Berechnung bestimmt werden müssen. Ein Beispiel für eine derartige Schrittlängenwahl ist in Schritt 5 des Algorithmus aus Beispiel (8.76) zu finden. Dieses modifizierte Subgradientenverfahren hat sich bei praktischen Versuchen bezüglich des symmetrischen TSP recht gut bewährt:

(8.76) Algorithmus (Subgradientenverfahren für symmetrisches TSP).
Eingabe: Symmetrisches TSP mit Zielfunktion c, $c_{ij} \geq 0$, $1 \leq i < j \leq n$.
Ausgabe: Untere Schranke L für den Optimalwert des TSP.

1. Initialisierung:
   $U :=$ obere Schranke für die kürzeste Tour,

$L := 0$ (gegenwärtig beste untere Schranke),
$\lambda^T = (\lambda_1, \ldots, \lambda_n) := (0, \ldots, 0)$ (Anfangsknotengewichte),
$0 < \alpha \leq 2$ (Skalierungsfaktor für die Schrittlänge, i. A. $\alpha := 2$),
$\varepsilon_1, \varepsilon_2, \varepsilon_3 > 0$ (numerische Genauigkeitsschranken),
$k \in \mathbb{N}$ (Zählparameter).

2. Berechne den kürzesten 1-Baum B bezüglich c^λ mit $c^\lambda_{ij} := c_{ij} + \lambda_i + \lambda_j \; \forall\, i, j$.
   Falls B eine Tour ist $\longrightarrow$ STOP.

3. Berechne einen Subgradienten u durch $u_i := d_B(i) - 2$ für $i = 1, \ldots, n$.

4. Setze $L := \max\{L, c(B) + \lambda^T u\}$ (gegenwärtig beste untere Schranke).
   - Falls $U - L < \varepsilon_1$ ist $\longrightarrow$ STOP (gewünschte Güte erreicht).
   - Falls sich L in den letzten k Schritten nicht wenigstens um ε_2 verbessert hat $\longrightarrow$ STOP. (Wir geben auf, weil das Verfahren stagniert. Weiterrechnen wäre nur Zeitverschwendung).

5. Setze
$$t := \alpha \frac{U - L}{\sum_{i=1}^n u_i^2} \qquad \text{(Schrittlänge)}.$$
   Ist $t < \varepsilon_3 \longrightarrow$ STOP. (Die Schrittlänge ist so klein geworden, dass numerisch nichts mehr passieren wird. Wir geben auf.)

6. Setze $\lambda := \lambda + tu$ und gehe zu Schritt 2. △

Die obere Schranke U in Schritt 1 erhält man z. B. durch Anwenden einer Primalheuristik oder man nimmt eine triviale obere Schranke, wie z. B.

$$U = n \cdot \max_{1 \leq i < j \leq n} c_{ij} \qquad \text{oder} \qquad U = \sum_{i=1}^{n} \max_{j \neq i} c_{ij}.$$

Wir haben zu Testzwecken $\alpha := 2$, $k := 100$ und $\varepsilon_1, \varepsilon_2, \varepsilon_3 := 10^{-3}$ gesetzt und für unser 6-Städte Beispiel das obige Programm ausgeführt. Das Subgradientenverfahren hat nach 276 Iterationen (d. h. 276 1-Baum Berechnungen) eine Tour geliefert. Damit ist diese Tour die optimale Lösung unseres 6-Städte Problems. Die optimalen Knotengewichte der 6 Städte sind in Abbildung 8.21 verzeichnet. (Diese Tour wurde auch durch einige unserer Primalheuristiken aus Abschnitt 8.1 gefunden.)

Es ist zu bemerken, dass die Optimallösung von (8.65) im Falle des TSP (und nicht nur hier) nicht unbedingt eine Tour liefern muss, sondern es können durchaus „gaps“ auftreten.

Trotzdem hat sich der Subgradientenansatz über die Lagrange-Relaxierung speziell für Probleme mittlerer Größenordnung bewährt. Verfahren dieser Art liefern i. A. sehr gute untere (bzw. bei Maximierungsproblemen obere) Schranken und kombiniert mit Branch & Bound gehören sie vielfach zu den besten Algorithmen, die wir für schwere kombinatorische Optimierung kennen.

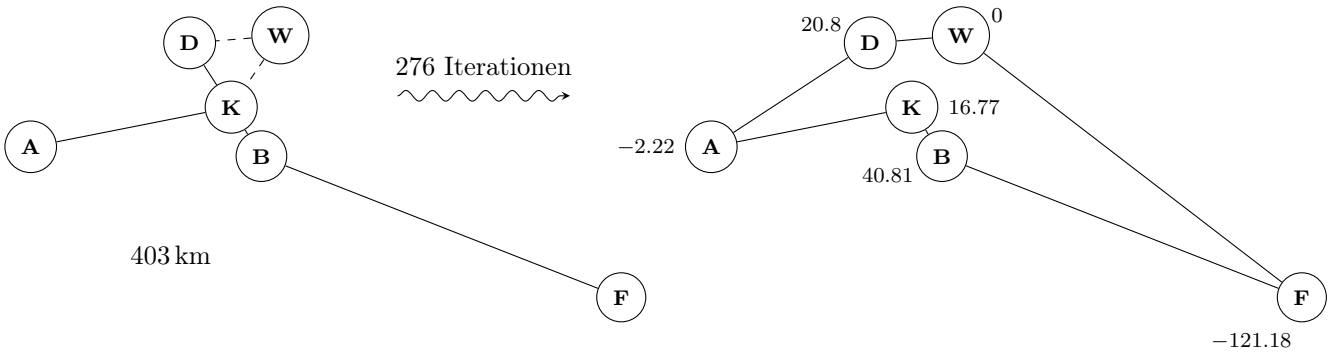


Abbildung 8.21: Ergebnis des Subgradientenverfahrens für das Rheinland

Zum Ende dieses Abschnittes möchten wir noch ein weiteres TSP-Beispiel vorführen und auf einige (numerische) Eigenschaften des Subgradientenverfahrens zur Lösung der Lagrange-Relaxierung hinweisen.

Die Werte (und damit unteren Schranken), die das Subgradientenverfahren liefert, hängen natürlich stark von den gewählten STOP-Kriterien ab. Je nachdem, wie man in dem Spezialverfahren für das TSP die Parameter k, ε_1, ε_2, ε_3, α setzt, wird man natürlich andere Resultate erhalten. Außerdem muss man bei der 1-Baum Relaxierung einen Knoten v spezifizieren. Obwohl theoretisch die Wahl von v irrelevant ist (für alle v ist $\max\{f(\lambda) \mid \lambda \in \mathbb{R}^n\}$ gleich), zeigt sich doch bei praktischen Rechnungen, dass unterschiedliche untere Schranken, abhängig von der Wahl, auftreten können. Wir demonstrieren das am folgenden Beispiel, von dem mir U Zaw Win berichtet hat.

(8.77) Beispiel (14-Städte Burma-Problem). Burma besteht aus 14 Bundesländern. Zur Erinnerung an den Freiheitskampf gegen England und zur Festigung der nationalen Einheit wird in jedem Jahr die Nationalflagge auf eine Rundreise durch die Hauptstädte der 14 Bundesländer geschickt. Diese Hauptstädte sind mit ihren geographischen Koordinaten in Tabelle 8.4 aufgelistet. Die Rundreise beginnt jeweils am 30. Januar eines jeden Jahres in der Bundeshauptstadt Rangoon, führt dann durch die übrigen 13 Städte und kehrt zum Nationalfeiertag am 12. Februar nach Rangoon zurück. Der Transport der Flagge erfolgt – je nach Verkehrsverbindungen – zu Fuß (Mandalay nach Sagaing), mit dem Schiff, dem Auto oder dem Flugzeug. Wir gehen davon aus, dass die Entfernung zwischen den Städten durch die Luftliniendistanzen auf der Erde gegeben sind, die man aus den Daten der Tabelle 8.4 berechnen kann; die so bestimmte Entfernungsmatrix findet sich in Tabelle 8.5. Wie sollte die Rundreise der Nationalflagge ausgelegt werden, so dass der Reiseweg möglichst kurz ist?

Für das Burma-Problem haben wir eine Rundreise mit Hilfe des Farthest-Insert-Algorithmus und dem Anfangskreis 7, 11, 14 berechnet. Diese Rundreise hat die Länge 3 322 km und ist in Abbildung 8.22(a) (nicht maßstabsgetreu) dargestellt. Um die Qualität dieser Lösung abzuschätzen, haben wir untere Schranken mit Hilfe des Subgradientenverfahrens berechnet.

Nr	Stadt	Breitengrad (N)	Längengrad (E)
1	Rangoon	16.47	96.10
2	Bassein	16.47	94.44
3	Sittwe	20.09	92.54
4	Haka	22.39	93.37
5	Myitkyina	25.23	97.24
6	Mandalay	22.00	96.05
7	Taunggyi	20.47	97.02
8	Pegu	17.20	96.29
9	Moulmein	16.30	97.38
10	Tavoy	14.05	98.12
11	Pa-an	16.53	97.38
12	Sagain	21.52	95.59
13	Loikaw	19.41	97.13
14	Magwe	20.09	94.55

Tabelle 8.4: 14 Städte in Burma (Geografische Koordinaten)

	2	3	4	5	6	7	8	9	10	11	12	13	14
1	153	510	706	966	581	455	70	160	372	157	567	342	398
2		422	664	997	598	507	197	311	479	310	581	417	376
3			289	744	390	437	491	645	880	618	374	455	211
4				491	265	410	664	804	1070	768	259	499	310
5					400	514	902	990	1261	947	418	635	636
6						168	522	634	910	593	18	284	239
7							389	482	757	439	163	124	232
8								154	406	133	508	273	355
9									276	43	623	358	498
10										318	898	633	761
11											582	315	464
12												275	221
13													247

Tabelle 8.5: 14 Städte in Burma (Distanzen auf der Erdoberfläche)

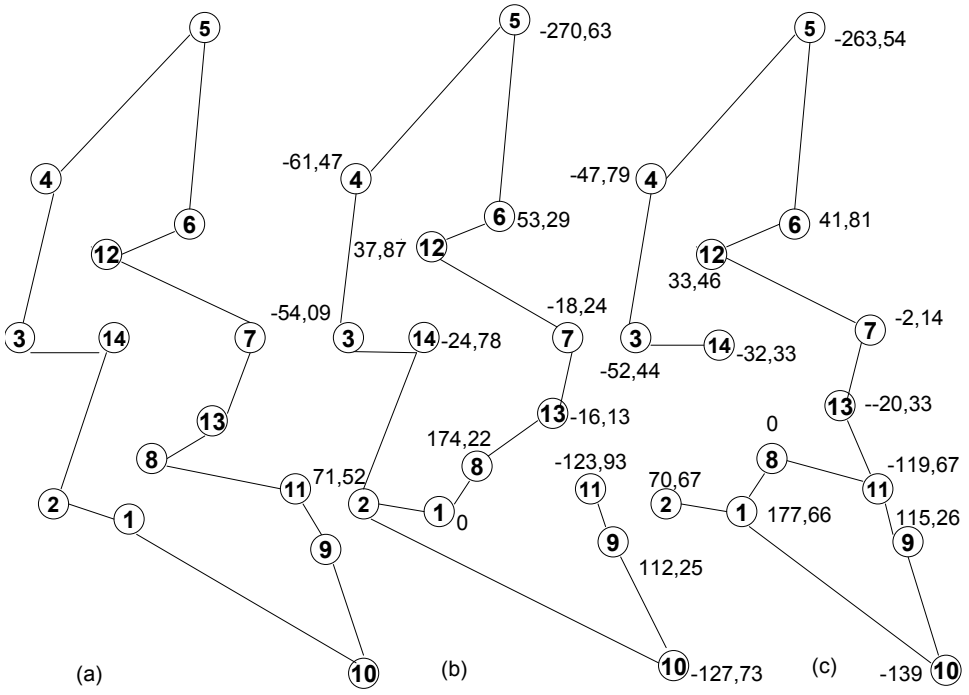


Abbildung 8.22: Lösungen zum Burma-TSP

Wählen wir als Knoten v des 1-Baumes den Knoten 1 (Rangoon), so erhalten wir den in Abbildung 8.22(b) gezeigten 1-Baum der Länge 3 313,58398 km. Die „optimalen“ Knotengewichte sind in Abbildung 8.22(b) an den Knoten eingetragen.

Wählen wir als Knoten v den Knoten 8 (Pegu), so erhalten wir einen 1-Baum der Länge 3 316,98 km. Er ist in Abbildung 8.22(c) dargestellt. Die „optimalen“ Knotengewichte sind dort ebenfalls verzeichnet.

Wir sehen also, dass bei unterschiedlicher Wahl des Knotens v in der 1-Baum-Relaxierung aufgrund unserer heuristischen Stopregeln verschiedene untere Schranken berechnet werden können. Da unser Burma-Problem ganzzahlige Daten hat, wissen wir, dass die kürzeste Rundreise nicht kürzer als 3 317 km sein kann. Der maximale absolute Fehler der von uns gefundenen Rundreise beläuft sich somit auf 5 km, dies entspricht einem maximalen relativen Fehler von etwa 0,15%. △

Schließlich wollen wir noch demonstrieren, dass nicht nur bei den bisher betrachteten kleinen „Spielbeispielen“ durch die Lagrange-Relaxierung gute Resultate erzielt werden, sondern auch bei großen. Erinnern wir uns an das 120-Städte Deutschland-Problem mit Optimallösung 6 942 km. Die (einfache) 1-Baum-Relaxierung, siehe Abbildung 8.16, brachte eine (nicht allzu gute) untere Schranke von 6 025 km (Knoten $v = 13 =$ Berlin).

Wir haben das Subgradientenverfahren aus (8.76) auf diese 1-Baum-Relaxierung angesetzt und laufen lassen, bis eines der STOP-Kriterien erfüllt war. Das Ergebnis ist in Abbildung 8.23 zu sehen. (An einigen Knoten sind die Lagrange-Multiplikatoren der letz-

ten („optimalen“) Lösung verzeichnet). Dieser 1-Baum hat eine Länge von 6 912 km. Die Abweichung dieses Wertes von der Länge der Optimaltour beträgt also nur noch 0,43%.

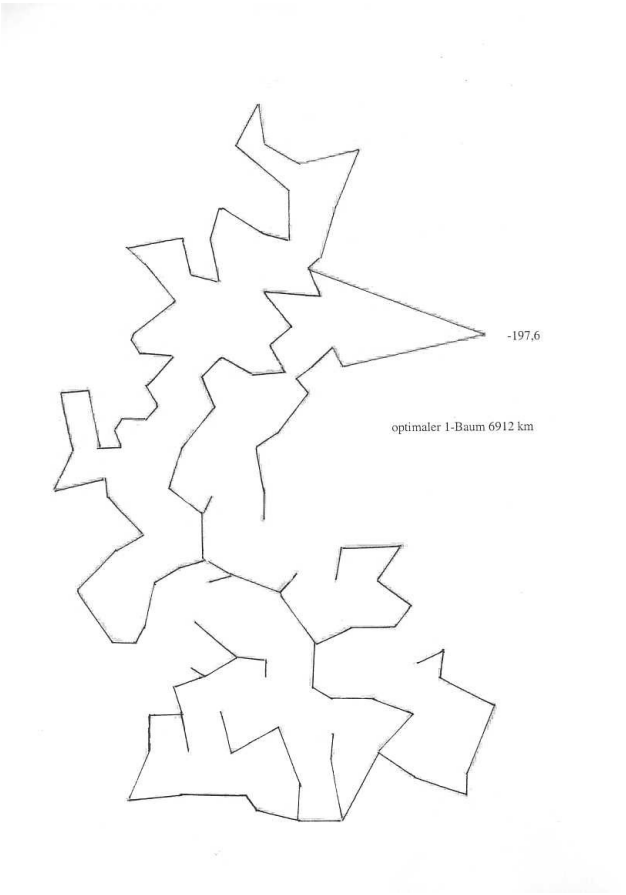


Abbildung 8.23: Besserer 1-Baum für das 120-Städte-Deutschland-TSP

Literaturverzeichnis

E. Aarts und J. K. Lenstra. *Local Search in Combinatorial Optimization.* Wiley, Chichester, first edition, 1997.

D. L. Applegate, R. E. Bixby, V. Chvátal, und W. J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, Princeton, 2006.

B. S. Baker. A new proof for the first-fit decreasing bin-packing algorithm. *Journal of Algorithms*, 6:47–70, 1985.

B. Chandra, H. Karloff, und C. Tovey. New results on the old k-opt algorithm for the traveling salesman problem. *SIAM Journal on Computing*, 28(6):1998–2029, 1999.

E. G. Coffman, M. R. Garey, und D. S. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, Kapitel 2, S. 46–93. PWS Publishing Company, Boston, 1996.

G. Dósa. The tight bound of first fit decreasing bin-packing algorithm is $FFD(I) \leq 11/9OPT(I) + 6/9$. In *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, Band 4614 von *Springer Lecture Notes in Computer Science*, S. 1–11. Springer, 2007.

G. Galambos und G. J. Woeginger. Repacking helps in bounded space online bin-packing. *Computing*, 22:349–355, 1993.

P. C. Gilmore, E. L. Lawler, und D. B. Shmoys. Well-solved special cases. In E. L. Lawler, J. K. Lenstra, A. Rinnooy Kan, und D. Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, S. 87–143. Wiley, 1985.

M. Grötschel. Schnelle Rundreisen: Das Travelling-Salesman-Problem. In S. Hußmann und B. Lutz-Westphal, editors, *Kombinatorische Optimierung erleben – In Studium und Unterricht*, Kapitel 4, S. 93–129. Vieweg, Wiesbaden, 2007. Elektronisch verfügbar unter `www.zib.de/groetschel/pubnew/paper/groetschel2007a_pp.pdf`.

M. Grötschel und M. Padberg. Die optimierte Odyssee. *Spektrum der Wissenschaft*, 4: 76–85, 1999.

M. Grötschel, S. O. Krumke, und J. Rambau. *Online Optimization of Large Scale Systems*. Springer, 2001.

G. Gutin und A. P. Punnen. *The Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, 2002.

D. S. Hochbaum und D. B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10:180–184, 1985.

W. L. Hsu und G. L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1:209–216, 1979.

D. S. Johnson und C. H. Papadimitriou. Performance guarantees for heuristics. In E. L. Lawler, J. K. Lenstra, A. Rinnooy Kan, und D. Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, S. 145–180. Wiley, 1985.

D. S. Johnson, C. R. Aragaon, L. A. McGeoch, und C. Schevon. Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning. *Operations Research*, 37(6):865–892, 1989.

D. S. Johnson, C. R. Aragaon, L. A. McGeoch, und C. Schevon. Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning. *Operations Research*, 39(3):378–406, 1991.

S. Kirkpatrick, C. D. Gelatt, und M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, und D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, 1985.

C. C. Lee und D. T. Lee. A simple online bin-packing algorithm. *Journal of the ACM*, 32:562–572, 1985.

S. Lin und B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

G. Reinelt. *The Traveling Salesman, Computational Solutions for TSP Applications*. Springer, 1994.

S. S. Seiden. On the online bin packing problem. In *Automata, Languages and Programming*, Band 2076 von *Springer Lecture Notes in Computer Science*, S. 237–248. Springer, 2001.

P. J. M. van Laarhoven und E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Co., Dordrecht, 1987.

A. van Vliet. An improved lower bound for on-line bin packing algorithms. *Information Processing Letters*, 43:277–284, 1992.