Kapitel 3

Lineare Optimierung

Ein Optimierungsproblem besteht für uns aus einem zulässigen Bereich $S \subseteq \mathbb{R}^n, \mathbb{Z}^n$ und einer Zielfunktion $c : S \to \mathbb{R}$. Ziel ist die Bestimmung von inf, sup, max oder $\min_{x \in S} c(x)$. Im allgemeinen sind solche Probleme beliebig schwer. Deswegen behandeln wir die Aufgabenstellung nicht in aller Allgemeinheit.

Beschränken wir uns zunächst auf den einfachsten Fall, daß $c(x) = c^\top x$ eine lineare Funktion ist und S eine Teilmenge des $\mathbb{R}^n$ ist, die durch eine Menge linearer Ungleichungen der Form $a^\top x \leq b$ gegeben ist.

3.1 Modellbildung

Lineare Programmierungsaufgaben aus der industriellen Praxis sind üblicherweise groß (tausende Variablen und Ungleichungen). Aus Gründen der Übersichtlichkeit behandeln wir hier eher "toy problems".

Beispiel 3.1.1 *Eine Ölraffinierie hat vier verschiede Sorten Rohbenzin zur Verfügung und mischt daraus Benzin in vier verschiedenen Oktanstärken. Dafür gelten folgende Daten*

Rohstoffsorte	*Oktanzahl*	*Fässer verfügbar*	*Preis pro Faß*
1	*68*	*4000*	$ *31.02*
2	*86*	*5050*	$ *33.15*
3	*91*	*7100*	$ *36.35*
4	*99*	*4300*	$ *38.75*

Benzinsorte	*Mindestoktanzahl*	*Nachfrage*	*Preis pro Faß*
1	*85*	$\geq$ *15000*	*\$ 40.99*
2	*90*	*beliebig*	*\$ 42.95*
3	*95*	$\leq$ *10000*	*\$ 45.15*

Gesucht ist ein Produktionsprozeß, der den Gewinn maximiert. Wir setzen

x_{ij} *:= Fässer des Rohstoffs i, die zur Produktion von Sorte j benutzt werden.*

Die Daten aus der ersten Tabelle liefern dann folgende Restriktionen.

$$\begin{aligned} x_{1,1}+x_{1,2}+x_{1,3} &\leq 4000 \\ x_{2,1}+x_{2,2}+x_{2,3} &\leq 5050 \\ x_{3,1}+x_{3,2}+x_{3,3} &\leq 7100 \\ x_{4,1}+x_{4,2}+x_{4,3} &\leq 4300 \end{aligned}$$

Auch die Anforderungen an die Qualität der Mischungen ergeben lineare Bedingungen, denn wir haben z.B.

$$\frac{68x_{1,1}+86x_{2,1}+91x_{3,1}+99x_{4,1}}{x_{1,1}+x_{2,1}+x_{3,1}+x_{4,1}} \geq 85.$$

Wir erhalten somit folgende drei Bedingungen

$$\begin{aligned} 68x_{1,1}+86x_{2,1}+91x_{3,1}+99x_{4,1}-85(x_{1,1}+x_{2,1}+x_{3,1}+x_{4,1}) &\geq 0 \\ 68x_{1,2}+86x_{2,2}+91x_{3,2}+99x_{4,2}-90(x_{1,2}+x_{2,2}+x_{3,2}+x_{4,2}) &\geq 0 \\ 68x_{1,3}+86x_{2,3}+91x_{3,3}+99x_{4,3}-95(x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}) &\geq 0 \end{aligned}$$

Der Mindest und Höchstabsatz ergeben:

$$\begin{aligned} x_{1,1}+x_{2,1}+x_{3,1}+x_{4,1} &\geq 15000 \\ x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3} &\leq 10000 \end{aligned}$$

Als Zielfunktion, die wir maximieren wollen, erhalten wir $c(x) = 40.99(x_{1,1}+x_{2,1}+x_{3,1}+x_{4,1})+42.95(x_{1,2}+x_{2,2}+x_{3,2}+x_{4,2})+45.15(x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3})-31.02(x_{1,1}+x_{1,2}+x_{1,3})-33.15(x_{2,1}+x_{2,2}+x_{2,3})-36.35(x_{3,1}+x_{3,2}+x_{3,3})-38.75(x_{4,1}+x_{4,2}+x_{4,3})$ *und bemerken abschließend, daß alle* $x_{ij} \geq 0$ *sein sollten.*

In Tabellenform erhalten wir also zunächst folgende Aufgabenstellung:

$x_{1,1}$	$x_{2,1}$	$x_{3,1}$	$x_{4,1}$	$x_{1,2}$	$x_{2,2}$	$x_{3,2}$	$x_{4,2}$	$x_{1,3}$	$x_{2,3}$	$x_{3,3}$	$x_{4,3}$		
9.97	7.84	4.64	2.24	11.93	9.80	6.60	4.20	14.13	12.00	8.80	6.40		
1	0	0	0	1	0	0	0	1	0	0	0	$\leq$	4000
0	1	0	0	0	1	0	0	0	1	0	0	$\leq$	5050
0	0	1	0	0	0	1	0	0	0	1	0	$\leq$	7100
0	0	0	1	0	0	0	1	0	0	0	1	$\leq$	4300
-17	1	6	14	0	0	0	0	0	0	0	0	$\geq$	0
0	0	0	0	-22	-4	1	9	0	0	0	0	$\geq$	0
0	0	0	0	0	0	0	0	-27	-9	-4	4	$\geq$	0
1	1	1	1	0	0	0	0	0	0	0	0	$\geq$	15000
0	0	0	0	0	0	0	0	1	1	1	1	$\leq$	10000

Bemerkung 3.1.2 *Bei richtigen Problemen ist die Modellbildung natürlich nicht so eindeutig wie in diesem Beispiel. Dabei können verschiedene Probleme auftauchen. Zunächst liegt ein realistisches Problem nicht in einer klar faßbaren mathematischen Form vor. Oft hat man konkurrierende Optimierungsziele und weiche Nebenbedingungen, die man vom Anwender oft nur erfährt, wenn er mittelit, warum ihm eine Lösung nicht gefällt.*

Zum anderen kann man mögliche Lösungsmengen durch unterschiedliche Ungleichungssysteme beschreiben, etwa indem man zusätzliche Variablen einführt. Je nach Formulierung sind die Probleme – speziell von einer bestimmten Software – schwerer oder leichter lösbar.

Zur allgemeinen Behandlung eines Optimierungsproblems ist es ungünstig, eine Mischung aus $\leq, \geq$ und $=$ Restriktionen zu haben. Außerdem wollen wir im Falle der Linearen Programmierung – wie im vorliegenden Beispiel – lieber Maximierungsprobleme als Minimierungsprobleme betrachten. Das macht keinen Unterschied, da $\max_{x \in S} c(x) = -(\min_{x \in S} -c(x))$. Deswegen definieren wir.

Definition 3.1.3 *Sei $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, b \geq 0$ und $c \in \mathbb{R}^n$. Ferner sei $m \leq n$ und A habe vollen Rang, also $rang(A) = n$. Die Aufgabenstellung*

$$\begin{array}{ll} \max & c^\top x \\ \textit{unter} & Ax = b \\ & x \geq 0 \end{array}$$

nennen wir Lineares Optimierungsproblem in Standardform. *Ist $x \geq 0$ mit $Ax = b$, so sagen wir x ist* zulässig *für das Problem.*

Um Beispiel 3.1.1 in Standardform zu bringen, führen wir für die neun Ungleichungen sogenannte *Schlupfvariabelen* $y_1, \ldots, y_9$ ein und setzen

$$A = \begin{pmatrix} 1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0\\ -17&1&6&14&0&0&0&0&0&0&0&0&0&0&0&0&-1&0&0&0&0\\ 0&0&0&0&-22&-4&1&9&0&0&0&0&0&0&0&0&0&-1&0&0&0\\ 0&0&0&0&0&0&0&0&-27&-9&-4&4&0&0&0&0&0&0&-1&0&0\\ 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1&0\\ 0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&1 \end{pmatrix}$$

$$b^\top = (4000, 5050, 7100, 4300, 0, 0, 0, 15000, 10000))$$

und $c^\top =$
(9.97,7.84,4.64,2.24,11.93,9.80 ,6.60,4.20,14.13,12.00,8.80,6.40,0,0,0,0,0,0,0,0,0).

Bemerkung 3.1.4 *Nicht vorzeichenbeschränkte Variablen u kann man mit $u = u^+ - u^-$ und $u^+, u^- \geq 0$ durch zwei vorzeichenbeschränkte ersetzen und so in Standardform bringen.*

Der älteste und immer noch praktisch wichtigste Algorithmus zur Lösung linearer Programme ist der Simplexalgorithmus. In den letzten zehn Jahren haben die innere-Punkt-Verfahren, die Ideen aus der Nichtlinearen Programmierung benutzen, an Bedeutung gewonnen.

Zur Lösung linearer Programme gibt es Standardpakete wie CPLEX, XPRESS oder OSL. Für nichtkommerzielle Belange gibt es auch hinreichend gute Software in der Public Domain. Ein Startpunkt ist die Electronic Library of Mathematical Software http://elib.zib.de.

CPLEX liefert als Lösung des obigen Problems:

```
CPLEX> display problem all
Minimize
 ZIEL: - 9.97 X11 - 11.93 X12 - 14.13 X13 - 7.84 X21 - 9.8 X22 - 12 X23
 - 4.64 X31 - 6.6 X32 - 8.8 X33 - 2.24 X41 - 4.2 X42 - 6.4 X43
Subject To
 RB1: X11 + X12 + X13 <= 4000
 RB2: X21 + X22 + X23 <= 5050
 RB3: X31 + X32 + X33 <= 7100
 RB4: X41 + X42 + X43 <= 4300
```

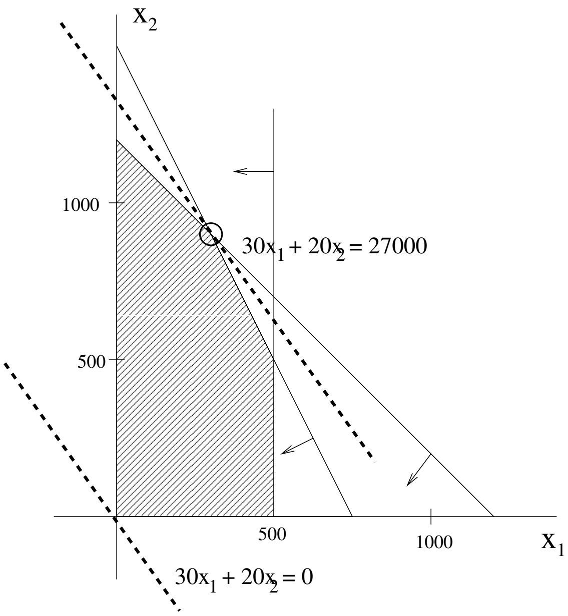
```
 O85: - 17 X11 + X21 + 6 X31 + 14 X41 >= 0
 O90: - 22 X12 - 4 X22 + X32 + 9 X42 >= 0
 O95: - 27 X13 - 9 X23 - 4 X33 + 4 X43 >= 0
 NF1: X11 + X21 + X31 + X41 >= 15000
 NF2: X13 + X23 + X33 + X43 <= 10000
Bounds
 All variables are >= 0.
CPLEX> optimize
Primal - Optimal:  Objective =   -1.3862560000e+05
Solution time =    0.00 sec.  Iterations = 10 (4)

CPLEX> display solution variables -
Variable Name           Solution Value
X11                        4000.000000
X21                        3823.846154
X23                        1226.153846
X31                        7100.000000
X41                        1541.153846
X43                        2758.846154
All other variables in the range 1-12 are zero.
```

Die algorithmische Idee des Simplexverfahrens und die Geometrie der Aufgabenstellung erkennt man am leichtesten an Beispielen wie dem folgenden:

Beispiel 3.1.5 (Graphische Lösung von Problemen mit 2 Variablen) *Eine Düngemittelfabrik stellt zwei Sorten Dünger A und B aus drei Ausgangsstoffen C,D,E her. Vom Ausgangstoff C stehen pro Periode 1500 Tonnen zur Verfügung, von D 1200 t und von E 500 t. Zur Produktion einer Tonne A werden 2 Tonnen C, und jeweils 1 Tonne D und E benötigt, für B jeweils eine Tonne C und D. Der Gewinn pro Tonne A beträgt 30,- DM, pro Tonne B 20,-. Wir erhalten also das Programm.*

$$\begin{array}{rrcrcr}
\max & 30x_1 & + & 20x_2 & & \\
\textit{unter} & 2x_1 & + & x_2 & \leq & 1500 \\
 & x_1 & + & x_2 & \leq & 1200 \\
 & x_1 & & & \leq & 500 \\
 & x_1 & , & x_2 & \geq & 0
\end{array}$$



Die Ungleichungen definieren jeweils einen Halbraum. Der zulässige Bereich ist als Schnitt von Halbräumen ein sogenanntes Polyeder. *Die Isogewinnflächen sind Hyperebenen (in der Ebene Geraden). Die Optimallösung erhält man, in dem man die Isogewinnfläche so weit wie möglich in Richtung wachsender Erlöse verschiebt, bis sie das Polyeder nur noch berührt. Betrachten wir die Ecken des Polyeders etwas genauer. Sie sind dadurch definiert, daß zwei Ungleichungen nicht strikt sind, sondern mit Gleichheit angenommen werden. Dies ist zunächst* $x_1 = 0, x_2 = 0$. *Wenn wir mit der Isogewinnfläche in Richtung wachsender Erlöse passieren wir* $(x_1 = 500, x_2 = 0), (x_1 = 500, 2x_2 + x_1 = 1500), (x_1 + x_2 = 1200, 2x_1 + x_2 = 1500)$. *Aus letzterem erhalten wir* $x_1 = 300, x_2 = 900$.

Bevor wir in Abschnitt 3.4 auf diese Idee zurückkommen, müssen wir den Korrektheitsbeweis mit etwas Theorie vorbereiten. Abschließend definieren wir Polyeder noch formal.

Definition 3.1.6 *Eine Menge* $P \subseteq \mathbb{R}^n$ *heißt Polyeder, wenn es ein* $m \in \mathbb{N}$, *eine* $\mathbb{R}^{m \times n}$ *Matrix* A *und ein* $b \in \mathbb{R}^m$ *gibt mit*

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\}.$$

Übung 3.1.7 *Zeigen Sie: Der zulässige Bereich eines linearen Programms in Standardform ist ein Polyeder.*

3.2 Farkas' Lemma

Lemma 3.2.1 (Farkas' Lemma) *Sei $L \subseteq \mathbb{R}^n$ ein Vektorraum. Dann gilt genau eine der beiden folgenden Altenativen:*

a) $\exists x \in L : x_1 > 0, x \geq 0$

b) $\exists y \in L^\perp : y_1 > 0, y \geq 0.$

Beweis. Offensichtlich können nicht beide Alternativen gleichzeitig gelten, da es ansonsten x, y gäbe mit $0 = x^\top y = \sum_{i=1}^n x_i y_i \geq x_1 y_1 > 0$.

Für den Rest der Behauptung führen wir vollstandige Induktion über n.

$n = 1$**:** Hier ist entweder $L = \mathbb{R}$ oder $L^\perp = \mathbb{R}$. Also gilt die Behauptung.

$n \geq 2$. Wir betrachten folgende Vektorräume(!) in $\mathbb{R}^{n-1}$

$$
\begin{aligned}
\tilde{L} &:= \{\tilde{x} \in \mathbb{R}^{n-1} \mid \exists x \in L, x_n = 0, x_i = \tilde{x}_i, i = 1, \ldots, n-1\} \\
\hat{L} &:= \{\hat{x} \in \mathbb{R}^{n-1} \mid \exists x \in L, x_i = \hat{x}_i, i = 1, \ldots, n-1\} \\
\tilde{L^\perp} &:= \{\tilde{y} \in \mathbb{R}^{n-1} \mid \exists y \in L^\perp, y_n = 0, y_i = \tilde{y}_i, i = 1, \ldots, n-1\} \\
\hat{L^\perp} &:= \{\hat{y} \in \mathbb{R}^{n-1} \mid \exists y \in L^\perp, y_i = \hat{y}_i, i = 1, \ldots, n-1\}
\end{aligned}
$$

Offensichtlich gilt dann $(\tilde{L})^\perp = \hat{L^\perp}$ und $(\hat{L})^\perp = \tilde{L^\perp}$. Falls es nun ein $\tilde{x} \in \tilde{L}$ mit $\tilde{x}_1 > 0, \tilde{x} \geq 0$ gibt, so leistet das zugehörige x das Gewünschte. Ebenso sind wir fertig, falls es ein $\tilde{y} \in \tilde{L^\perp}$ gibt mit $y_1 > 0, y \geq 0$. Wir können also davon ausgehen, daß jeweils die andere Alternative gilt, d.h. es gibt $x \in L, y \in L^\perp$ mit $x_1, y_1 > 0$ und $x_i, y_i \geq 0$ für $i = 1, \ldots, n-1$. Aus $x^\top y = 0$ erhalten wir somit

$$
\begin{aligned}
-x_n y_n &= x_1 y_1 + \sum_{i=2}^{n-1} x_i y_i \\
&\geq x_1 y_1 \\
&> 0.
\end{aligned}
$$

Wir schließen, daß genau einer von x_n und y_n echt positiv ist. Falls $x_n > 0$ ist, so leistet x das Gewünschte für die erste Alternative, im anderen Fall y für die zweite. □

Farkas' Lemma ist in einigen anderen Erscheinungsformen bekannt. Am häufigsten wird es wohl zitiert als

> Ist $P \subseteq \mathbb{R}^n$ ein Polytop und $x \in \mathbb{R}^n$, so gilt entweder $x \in P$ oder es gibt eine affine Hyperebene, die x und P trennt.

Ein Polytop ist ein beschränktes Polyeder, das in diesem Satz als *konvexe Hülle* von Punkten gegeben ist und nicht als Schnitt von Halbräumen. Die beiden Polyederbegriffe sind äquivalent, was wir im Rahmen dieser Vorlesung nicht beweisen werden. Bevor wir diese Version von Farkas' Lemma herleiten, erinnern wir zunächst an eine Tatsache aus der Linearen Algebra und definieren die konvexe Hülle.

Proposition 3.2.2 *Sei $A \in \mathbb{R}^{m\times n}$. Dann sind*

$$L = \mathit{ker}(A) := \{x \in \mathbb{R}^n \mid Ax = 0\}$$

und

$$\mathit{im}(A^\top) := \{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^m : x = A^\top y\}$$

ein orthokomplementäres Paar von Unterräumen des $\mathbb{R}^n$, d.h. $\mathit{im}(A^\top) = L^\perp$.

Definition 3.2.3 *Seien $x_1, \ldots, x_k \in \mathbb{R}^n$ und $\lambda \in \mathbb{R}^k$. Dann heißt*

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

Linearkombination *von $x_1, \ldots, x_k$. Gilt zusätzlich*

$$\left\{ \begin{array}{ll} \lambda \geq 0 & \\ & \sum_{i=1}^k \lambda_i = 1 \\ \lambda \geq 0, & \sum_{i=1}^k \lambda_i = 1 \end{array} \right\}, \textit{ so heißt } x \left\{ \begin{array}{l} \text{konische} \\ \text{affine} \\ \text{Konvex-} \end{array} \right\} \text{Kombination.}$$

Die Kombination heißt echt, wenn es $\lambda_i \neq 0 \neq \lambda_j$ mit $x_i \neq x_j$ gibt, oder der einzige Nichtnulleintrag von λ, etwa λ_i, von Eins verschieden ist und $x_i \neq 0$ ist. $S \subseteq \mathbb{R}^n$ sei

$$\left\{ \begin{array}{l} \mathit{lin}(S) \\ \mathit{cone}(S) \\ \mathit{aff}(S) \\ \mathit{conv}(S) \end{array} \right\}, \textit{ die Menge aller } \left\{ \begin{array}{l} \text{linearen} \\ \text{konischen} \\ \text{affinen} \\ \text{konvexen} \end{array} \right\} \textit{Kombinationen von}$$

Elementen in S. Wir sprechen von der Hülle.

Kommen wir also zum angekündigten Trennungssatz.

Korollar 3.2.4 *Seien $v_1, \dots, v_k, x \in \mathbb{R}^n$. Dann gilt:*
Entweder $x \in \operatorname{conv}(\{v_1, \dots, v_k\})$ oder es gibt $a \in \mathbb{R}^n, b \in \mathbb{R}$, so daß $a^\top v_i \leq b$ für $i = 1, \dots, k$ und $a^\top x > b$.

Beweis. Wir setzen

$$A = \begin{pmatrix} -x & v_1 & \dots & v_k \\ -1 & 1 & \dots & 1 \end{pmatrix}$$

und wenden Farkas' Lemma auf das Paar $L = \ker(A), L^\perp = \operatorname{im}(A)$ an. Dann gilt: entweder es gibt $(\mu_0, \mu) \in \mathbb{R}^{n+1}$, $\mu \geq 0, \mu_0 > 0$ mit $A\mu = 0$ oder es gibt ein $\binom{a}{a_{k+1}} \in \mathbb{R}^{k+1}$ mit $A^\top a \geq 0$ und $(A^\top \binom{a}{a_{k+1}})_1 > 0$. Letzteres bedeutet aber $(-a)^\top x > -a_{k+1}$ und $a^\top v_i \geq a_{k+1}$, also $(-a)^\top v_i \leq -a_{k+1}$ für $i = 1, \dots, k$. Im ersten Fall setzen wir $\lambda_i := \frac{\mu_i}{\mu_0}$ und erhalten $-x + \sum_{i=1}^k \lambda_i v_i = 0$ und $-1 + \sum_{i=1}^k \lambda_i = 0$. □

3.3 Dualitätssatz

Wir wollen nun die Resultate aus dem letzten Kapitel auf die Lineare Programmierung anwenden. Dafür leiten wir zunächst eine „gute Charakterisierung" wie folgt her. Betrachten wir die Behauptung

> Das Lineare Programm
>
> $$(P) \qquad \begin{array}{rc} \max & c^\top x \\ \text{unter} & Ax = b \\ x \geq 0 & \end{array}$$
>
> hat mindestens einen Wert von z_0.

Die Richtigkeit dieser Behauptung läßt sich durch Angabe eines $x \geq 0$ mit $Ax = b$ sowie $c^\top x \geq z_0$ beweisen. Wenn es kein solches x gibt, hilft uns Farkas' Lemma, dafür einen Beweis anzugeben. Betrachten wir nämlich den Kern der Matrix.

$$B := \begin{pmatrix} -b & A & 0 \\ -z_0 & c^\top & -1 \end{pmatrix}$$

so enthält dieser kein nichtnegatives Element, das in der ersten Komponente echt positiv ist. Denn wäre (x_0, x, x_{n+1}) ein solches Element, so haben wir $A\frac{x}{|x_0|} = b$, $c^\top \frac{x}{|x_0|} \geq z_0$. Nach Farkas' Lemma gibt es also ein (y, t) mit $y^\top b + tz_0 < 0, y^\top A + tc^\top \geq 0, t \leq 0$. Dies motiviert folgende Definition:

Definition 3.3.1 *Das Lineare Programm*

$$(D) \qquad \begin{array}{ll} \min & y^\top b \\ \textit{unter} & y^\top A \geq c^\top \end{array}$$

heißt das duale Programm *zum Linearen Programm in Standardform.*

Wir werden im folgenden zeigen, daß, im wesentlichen, das duale und primale Programm den gleichen Zielfunktionswert haben. Dies hilft beim Algorithmendesign.

Lemma 3.3.2 (Schwache Dualität) *Ist $x \geq 0$ zulässig für das primale Programm und y für das duale Programm, so gilt $c^\top x \leq y^\top b$.*

Beweis.

$$c^\top x \overset{x \geq 0}{\leq} y^\top Ax = y^\top b.$$

□

Satz 3.3.3 (Dualitätssatz der Linearen Programmierung) *Seien $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ und $c \in \mathbb{R}^n$. Dann gilt genau eine der folgenden vier Alternativen:*

a) *Das primale Programm ist zulässig und beschränkt. Ist dann z_0 eine kleinste obere Schranke, so ist auch das duale Programm zulässig und beschränkt und es gibt eine Optimallösung y^* mit $z_0 = y^{*\top} b$.*

b) *Das primale Programm ist unbeschränkt, d.h. es gibt ein $x_0 \geq 0$ mit $Ax_0 = b$ und ein $x_1 \geq 0$ mit $Ax_1 = 0$ und $c^\top x_1 > 0$.*

c) *Das duale Programm ist unbeschränkt, d.h. es gibt ein y_0 mit $y_0^\top A \geq c$ und ein y_1 mit $y_1^\top A \geq 0$ und $y_1^\top b < 0$.*

d) *Beide Programme sind unzulässig, d.h. es gibt weder ein $x \geq 0$ mit $Ax = b$, noch ein $y \in \mathbb{R}^m$ mit $y^\top A \geq c^\top$.*

Beweis. Wegen Lemma 3.3.2 kann stets höchstens eine Alternative gelten. Sei zunächst das primale Programm zulässig. Ist es unbeschränkt, so gilt Alternative b). Also nehmen wir an, das primale Programm sei beschränkt, sei dann z_0 eine kleinste obere Schranke. Da z_0 obere Schranke ist, gibt es kein $x \geq 0$ mit $Ax - b = 0$ und $c^\top x > z_0$. Wir betrachten den Kern von

$$B := \begin{pmatrix} 0 & A & -b \\ 1 & -c^\top & z_0 \end{pmatrix}.$$

Falls $(w_0, w^\top, w_{n+1})$ im Kern von B existiert mit $w_0 > 0, w \geq 0, w_{n+1} \geq 0$, so impliziert die Schrankeneigenschaft von z_0 sofort $w_{n+1} = 0$. Denn ansonsten wäre $x := \frac{1}{w_{n+1}} w$ zulässig für das primale Programm und $c^\top x > z_0$. Daraus folgt aber $Aw = 0$ und $c^\top w = w_0 > 0$, d.h. das primale Programm ist unbeschränkt im Widerspruch zur Annahme. Nach Farkas' Lemma gibt es also ein $(\tilde{y}, y_{m+1})$ mit $y_{m+1} > 0, \tilde{y}^\top A - c^\top y_{m+1} \geq 0, y_{m+1} z_0 \geq \tilde{y}^\top b$. Nach Division durch y_{m+1} haben wir somit ein y mit $y^\top A \geq c$ und $y^\top b \leq z_0$. Da z_0 kleinste obere Schranke ist und wegen Lemma 3.3.2 $y^\top b$ obere Schranke ist, gilt ebenso $y^\top b \geq z_0$.

Ist nun das primale Programm unzulässig, betrachten wir $(-b, A)$ und Farkas' Lemma impliziert die Existenz eines y mit $y^\top b < 0$ und $y^\top A \geq 0$. Also ist das duale Programm, falls es zulässig ist, unbeschränkt. □

Daß auch die letzte Alternative eintreten kann, zeigt folgendes Beispiel:

Beispiel 3.3.4

$$
\begin{array}{rlrcrcl}
 & \max & x_1 & + & x_2 & & \\
(P) & \textit{unter} & x_1 & - & x_2 & = & 1 \\
 & & -x_1 & + & x_2 & = & 1 \\
 & & x_1 & , & x_2 & \geq & 0
\end{array}
$$

$$
\begin{array}{rlrcrcl}
 & \min & y_1 & + & y_2 & & \\
(D) & \textit{unter} & y_1 & - & y_2 & \geq & 1 \\
 & & -y_1 & + & y_2 & \geq & 1
\end{array}
$$

Übung 3.3.5 *Zeigen Sie: Das duale Programm des dualen Programms ist das primale Programm. (Bringen Sie das duale Programm in Standardform und dualisieren Sie.) Folgern Sie hieraus: Ist das primale Programm zulässig und beschränkt, so gibt es für beide Programme Optimallösungen x^* und y^* und es gilt: $c^\top x^* = y^{*\top} b$. Diskutieren Sie den Unterschied zu Satz 3.3.3 a).*

Haben wir nun ein duales Paar linearer Programme, bei der die erste Alternative von Satz 3.3.3 gilt, so betrachten wir ein duales Paar von Optimallösungen x^*, y^*. Es ist $(y^*)^\top b - c^\top x^* = (y^{*\top} A - c^\top) x^* = 0$. Da $x^* \geq 0$ und $y^{*\top} A - c^\top \geq 0$, schließen wir hieraus den

Satz 3.3.6 (Satz vom komplementären Schlupf) *Seien $x^* \in \mathbb{R}^n_+$ mit $Ax = b$ und $y^* \in \mathbb{R}^m$ mit $y^{*\top} A \geq c^\top$. Dann sind x^*, y^* genau dann ein Paar von Optimallösungen des primalen bzw. dualen Programmes, wenn gilt:*

a) $x_i^* \neq 0 \Rightarrow (c^\top = y^{*\top}A)_i$ *und*

b) $(c^\top > y^{*\top}A)_i \Rightarrow x_i^* = 0.$

Beweis. Die Notwendigkeit der Bedingung hatten wir vor Formulierung des Satzes hergeleitet. Aus den Bedingungen folgt nun

$$c^\top x^* = \sum_{i=1}^n c_i x_i^* = \sum_{i=1}^n (y^{*\top}A)_i x^* i = y^{*\top}Ax^* = y^{*\top}b$$

also folgt aus Lemma 3.3.3 die Behauptung.

3.4 Das Simplexverfahren

Die geometrische Idee des Simplexverfahrens hatten wir im zweidimensionalen Fall schon erläutert. Wir gehen so lange von einer Ecke zu einer besseren Ecke, bis es nicht mehr besser geht. Ecken des Standardprogramms erhalten wir dadurch, daß wir Ungleichungen zu Gleichungen machen, hier also durch Nullsetzen möglichst vieler Variablen.

Definition 3.4.1 *Sei P ein Polyeder. Dann heißt $x \in P$* Ecke von *P, wenn x nicht echte Konvexkombination von Elementen in P ist.*

Lemma 3.4.2 *Habe $A \in \mathbb{R}^{m\times n}$ vollen Zeilenrang und seien $b \in \mathbb{R}^m, x \in \mathbb{R}^n, x \geq 0$. Dann ist x Ecke von $P := \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$ genau dann, wenn es $B \subseteq \{1,\dots,n\}$ gibt, mit $A_{.B}$ regulär, $x_B = A_{.B}^{-1}b$ und mit $N := \{1,\dots,n\} \setminus B$ gilt $x_N = 0$.*

Beweis. Seien $x^1,\dots,x^k \in P, \lambda_1,\dots,\lambda_k > 0, \sum_{i=1}^n \lambda_i = 1$ mit $x = \sum_{i=1}^n \lambda_i x_i$. Da die $x^i \geq 0$ und die $\lambda_i > 0$ sind, folgt aus $x_N = 0$, daß $x_N^i = 0$ für alle i. Aus $A_{.B}x_B^i = Ax_i = b$ schließen wir somit $x_B^i = A_{.B}^{-1}b = x_B$ für alle i. Also kann die Konvexkombination nicht echt sein und x ist eine Ecke. Nehmen wir nun für die andere Richtung der Aussage an, daß es ein solches B nicht gibt. Sei C die Menge der Indizes mit $x_C > 0$. Da C nicht zu einem B ergänzt werden kann, hat die Matrix $A_{.C}$ nicht vollen Spaltenrang. Also gibt es $0 \neq y_C \in \ker(A_{.C})$ und ein $\varepsilon > 0$ mit $x_C^1 := x_C + \varepsilon y_C > 0$ und $x_C^2 := x_C - \varepsilon y_C > 0$. Wir setzen $x^1_{\{1,\dots,n\}\setminus C} = 0 = x^2_{\{1,\dots,n\}\setminus C}$. Dann ist $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$ keine Ecke. □

Definition 3.4.3 *Habe $A \in \mathbb{R}^{m\times n}$ vollen Zeilenrang und sei $b \in \mathbb{R}^m$. Wir sagen $B \subseteq \{1,\dots,n\}$ ist eine* Basis *von A, falls $A_{.B}$ regulär ist, B heißt* zulässige Basis, *wenn $A_{.B}^{-1}b \geq 0$ ist. Ist B eine (zulässige) Basis, so heißt $N := \{1,\dots,n\} \setminus B$ (zulässige) Nichtbasis und der Vektor $x \in \mathbb{R}^n$ mit $x_B = A_{.B}^{-1}b, x_N = 0$ (zulässige) Basislösung.*

Als nächstes werden wir untersuchen, wie man von einer zulässigen Ecke eine benachbarte bessere findet oder feststellt, daß die Ecke eine Optimallösung darstellt.

Proposition 3.4.4 (Optimalitätskriterium) *Habe $A \in \mathbb{R}^{m\times n}$ vollen Rang und sei $b \in \mathbb{R}^m$. Sei B eine zulässige Basis. Dann ist die zulässige Basislösung x eine Optimallösung des linearen Programms, wenn*

$$c^\top - c_B^\top A_{.B}^{-1}A \leq 0.$$

Beweis. Wir setzen $y^\top = c_B^\top A_{.B}^{-1}$. Dann ist $y^\top A \geq c^\top$ und $y^\top b = c_B^\top A_{.B}^{-1}b = c_B^\top x_B = c^\top x$ und die Behauptung folgt aus der schwachen Dualität. □

Der Term $c^\top - c_B^\top A_{.B}^{-1}A$ liefert nicht nur eine Optimalitätskriterium sondern, wie wir gleich sehen werden, auch Information darüber, welche Nachbarecken lohnend sind.

Definition 3.4.5 *Ist B eine zulässige Basis, so heißen $c^\top - c_B^\top A_{.B}^{-1}A$* die reduzierten Kosten.

Der Begriff reduzierte Kosten kommt daher, daß man früher zuerst Minimierungsprobleme eingeführt hat. In unserem Falle spräche man besser von reduzierten Gewinnen.

Definition 3.4.6 *Zwei Basen B_1, B_2 heißen benachbart, wenn $|B_1 \triangle B_2| = 2$.*

Lemma 3.4.7 *Seien B und $B' = B \cup \{j\} \setminus \{i\}$ zwei benachbarte zulässige Basen mit zugehörigen Basislösungen x und x'. Dann ist*

$$c^\top x' - c^\top x = x'_j(c_j - c_B^\top A_{.B}^{-1}A_{.j}).$$

Beweis. Wegen $A(x' - x) = 0$ sind die Nichtnulleinträge von $x' - x$ im Kern von $A_{.,B'\cup B}$, wobei $(x'-x)_j = x'_j$ und $(x'-x)_i = -x_i$. Also ist $-x'_jA_{.j} = A_{.B}(x'-x)_B$ und somit $(x'-x)_B = -x'_j A_{.B}^{-1} A_{.j}$. Somit erhalten wir

$$\begin{aligned} c^\top x' - c^\top x &= c^\top_{B'\cup B}(x'-x)_{B'\cup B} \\ &= c_j x'_j + c_B^\top (x'-x)_B \\ &= x'_j c_j - x'_j c_B^\top A_{.B}^{-1} A_{.j} \\ &= x'_j (c_j - c_B^\top A_{.B}^{-1} A_{.j}). \end{aligned}$$

□

Wenn x'_j und die j-ten reduzierten Kosten beide echt positiv sind, verbessern wir uns beim Wechsel von x nach x'. Haben also die reduzierten Kosten noch einen positiven Eintrag, so wollen wir den zugehörigen Index, etwa j einer solchen Variablen in die Basis B aufnehmen. Im Kern der Matrix $A_{.,B\cup j}$ gibt es genau eine vom Nullvektor verschiedene Richtung, "nämlich $e_j - A_{.B}^{-1} A_{.j}$". Diese verläuft von der Basislösung in Richtung einer *Kante* des Polyeders. In dieser Richtung wird die Variable x_j von Null auf einen positiven Wert gehoben, bis eine Bedingung $x_i \geq 0$ greift. Da x_i auf 0 gesetzt wird, nennen wir es das *basisverlassende Element*. Dies ist also der Index, bei dem bei Erhöhungen der neuen Basisvariablen der zugehörige x-Wert als erstes auf Null, fällt:

Proposition 3.4.8 *Ist B eine zulässige Basis, $i \in B \not\ni j$ und*

$$i \in argmin \left\{ \frac{(A_{.B}^{-1} b)_i}{(A_{.B}^{-1} A_{.j})_i} \mid (A_{.B}^{-1} A_{.j})_i > 0 \right\},$$

wobei argmin die Menge aller Indizes ist, an denen das Minimum angenommen wird, so ist $B' := B \cup \{j\} \setminus \{i\}$ eine zulässige Basis.

Beweis. Ist $x = (x_B, x_N) = (x_B, 0)$ zulässige Basislösung, $j \notin B$ und d definiert durch

$$d_k := \begin{cases} -A_{.B}^{-1} A_{.j} & \text{falls} \quad k \in B \\ 1 & \text{falls} \quad k = j \\ 0 & \text{sonst,} \end{cases}$$

so gilt für alle $\lambda : A(x + \lambda d) = b + \lambda(Ae_j - A_{.j}) = b$. Ist nun

$$\lambda_0 = \min\{\frac{(A_{.B}^{-1} b)_i}{(A_{.B}^{-1} A_{.j})_i} \mid (A_{.B}^{-1} A_{.j})_i > 0\},$$

so gilt außerdem für alle $k \in B$:

$$\begin{aligned}(x+\lambda_0 d)_k &= (A_{.B}^{-1}b)_k + \lambda_0(-A_{.B}^{-1}A_{.j})_k \\ &\geq (A_{.B}^{-1}b)_k + \frac{(A_{.B}^{-1}b)_k}{(A_{.B}^{-1}A_{.j})_k}(-A_{.B}^{-1}A_{.j})_k = 0.\end{aligned}$$

Es bleibt zu zeigen, daß $A_{.B'}$ vollen Rang hat. Angenommen dies wäre nicht der Fall und $\tilde{\mu} \neq 0$ mit $A_{.B'}\tilde{\mu} = 0$. Sei dann μ definiert durch $\mu_{B'} = \tilde{\mu}$ und $\mu_{N'} = 0$. Dann gilt $\kappa := \mu - \mu_j d \neq 0$, denn $\kappa_i = -(A_{.B}^{-1}A_{.j})_i < 0$ und $\kappa_j = 0$. Somit definieren die Nichtnulleinträge von κ ein nichttriviales Element im Kern von $A_{.B}$ im Widerspruch dazu, daß B eine Basis ist. □

Bevor wir den Simplexalgorithmus formulieren, überlegen wir zunächst, was passiert, wenn die Menge, über die wir durch diese Minimumbildung das basisverlassende Element finden wollen leer ist. Anschaulich heißt das, daß nie mehr eine Bedingung $x \geq 0$ greift, wir also beliebig viel weiteren Profit einstreichen können.

Lemma 3.4.9 *Ist B eine zulässige Basis mit Basislösung x, $c_j - c_B^\top A_{.B}^{-1}A_{.j} > 0$ und $(A_{.B}^{-1}A_{.j})_i \leq 0$ für alle i, so ist das lineare Programm unbeschränkt.*

Beweis. Sei $d \in \mathbb{R}^n$ mit $d_j = 1, d_B = -A_{.B}^{-1}A_{.j}$ Dann ist für $\lambda \geq 0 : x + \lambda d \geq 0, A(x+\lambda d) = b$ und $c^\top(x+\lambda d) = c^\top x + \lambda(c_j - c_B^\top A_{.B}^{-1}A_{.j}) \overset{\lambda\to\infty}{\to} \infty$. □

Wir haben jetzt fast alle Fakten beisammen, um den Simplexalgorithmus skizzieren und beweisen zu können. Wir beschränken die Eingabedaten allerdings zunächst noch auf den günstigen Fall, daß A vollen Zeilenrang hat und eine zulässige *Startbasis B* gegeben ist. Wie man den allgemeinen Fall darauf reduziert, werden wir später vorstellen.

Schematische Skizze des Simplexalgorithmus:

Eingabedaten: $A \in \mathbb{R}^{m\times n}$ mit vollem Zeilenrang, $b \in \mathbb{R}^m, b \geq 0$, zulässige Basis B, $c \in \mathbb{R}^n$.

Solange $c^\top - c_B^\top A_{.B}^{-1}A \not\leq 0$:

Spaltenwahl: Wähle j mit $c_j - c_B^\top A_{.B}^{-1}A_{.j} > 0$.

Zeilenwahl: Berechne $i \in \operatorname{argmin}\{\frac{(A_{.B}^{-1}b)_i}{(A_{.B}^{-1}A_{.j})_i} \mid (A_{.B}^{-1}A_{.j})_i > 0\}$. Falls diese Menge leer ist. STOP. Das Programm ist unbeschränkt.

Basiswechsel: Setze $B = B \setminus \{i\} \cup \{j\}$.

oder als Pythoncode

```
def simplex(c,A,b,B):
  m=A.shape[0]
  n=A.shape[1]
  C=concatenate((A,b),1)     # 1 heisst nebeneinander
  c=concatenate((c,[[0]]),1)
  D=concatenate((c,C),0)     # 0 heisst untereinander
  initialize_basis(D,B)
  redcostneg=0
  while redcostneg==0:
    column=choosecolumn(D)
    if column<=n:
      row=chooserow(D,column)
      if row==0:
        print "Programm ist unbeschraenkt"
        redcostneg=1
      else:
        gaussjordanelim(D,row,column)
        B[row-1]=column
    else:
      redcostneg=1
  return D,B
```

3.5 Tableauform des Simplexalgorithmus

Alle die oben angeführten Operationen lassen sich mit Hilfe des Gauß-Jordan Eliminationsschrittes sehr leicht in einer Tableauform formalisieren. Wir nehmen wieder an, daß A vollen Zeilenrang hat und eine zulässige *Startbasis* B gegeben ist. Dann lautet das Tableau zur Basis B:

$$\begin{array}{c|c} c^\top - c_B^\top A_{.B}^{-1}A & -c_B^\top A_{.B}^{-1}b \\ \hline A_{.B}^{-1}A & A_{.B}^{-1}b \end{array}$$

Der Basiswechsel von B nach $B \setminus \{i\} \cup \{j\}$ wird nun mittels eines Gauß-Jordan Eliminationsschrittes mit Pivotelement $(A_{.B}^{-1}A)_{ij}$ durchgeführt. Als Beispiel wollen wir die Düngemittelfabrik durchrechnen. Startbasis sind hier die drei Schlupfvariablen y_1, y_2, y_3.

$$\begin{array}{ccccc|c} x_1 & x_2 & \underline{y_1} & \underline{y_2} & \underline{y_3} & -ZF \\ 30 & 20 & 0 & 0 & 0 & 0 \\ \hline 2 & 1 & 1 & 0 & 0 & 1500 \\ 1 & 1 & 0 & 1 & 0 & 1200 \\ \boxed{1} & 0 & 0 & 0 & 1 & 500 \end{array} \qquad \begin{array}{ccccc|c} \underline{x_1} & x_2 & \underline{y_1} & \underline{y_2} & y_3 & ZF \\ 0 & 20 & 0 & 0 & -30 & -15000 \\ \hline 0 & \boxed{1} & 1 & 0 & -2 & 500 \\ 0 & 1 & 0 & 1 & -1 & 700 \\ 1 & 0 & 0 & 0 & 1 & 500 \end{array}$$

$\underline{x_1}$	$\underline{x_2}$	y_1	$\underline{y_2}$	y_3	$-ZF$
0	0	−20	0	10	−25000
0	1	1	0	−2	500
0	0	−1	1	$\boxed{1}$	200
1	0	0	0	1	500

$\underline{x_1}$	$\underline{x_2}$	y_1	y_2	$\underline{y_3}$	ZF
0	0	−10	−10	0	−27000
0	1	−1	2	0	900
0	0	−1	1	1	200
1	0	1	−1	0	300

Bemerkung 3.5.1 *In einigen Lehrbüchern wird das Tableau für Minimierungsprobleme eingeführt, dann wählt man Spalten mit negativen reduzierten Kosten, bis alle positiv sind. Auch wird oft die Zielfunktion wird unter die Matrix geschrieben.*

3.6 Pivotwahl, Entartung, Endlichkeit

Die oben angegebene Skizze ist strenggenommen noch kein Algorithmus, da, insbesondere bei der Spaltenwahl noch viel Freiheit herrscht. Auch bei der Zeilenwahl gibt es bei nicht eindeutigem Minimum Zweideutigkeiten.

Eine feste Vorschrift der Auswahl der Pivotspalte bezeichnet man als *Pivotregel*. Wir wollen hier drei erwähnen.

Steilster Anstieg: Wähle die Spalte mit den größten reduzierten Kosten.

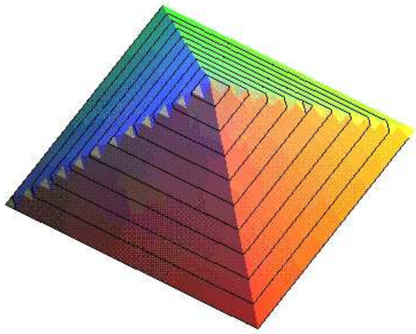
Größter Fortschritt: Wähle die Spalte deren Aufnahme in die Basis die größte Verbesserung in der Zielfunktion verspricht.

Bland's rule: Wähle stets die Variable mit dem kleinsten Index (sowohl bei Spalten als auch bei Zeilenwahl).

Die erste Regel ist billig zu implementieren mit zufriedenstellendem Ergebnis, die zweite Regel ist numerisch aufwendig und die dritte ist praktisch fast bedeutungslos hat aber ein wesentliches theoretisches Feature. Das liegt daran, daß ein Basiswechsel nicht notwendig zu einem Wechsel der Ecke führt.

Definition 3.6.1 *Eine Ecke x eines LP in Standardform heißt* entartet, *wenn es mindestens zwei Basen $B \neq B'$ gibt mit $x_B = A_{.B}^{-1}b$ und $x_{B'} = A_{.B'}^{-1}b$, d.h. die zugehörigen Basislösungen zu B und B' sind gleich. Ebenso nennen wir einen Pivotschritt* entartet, *der die Basislösung nicht verändert.*

Geometrisch liegt eine entartete Ecke auf mehr Hyperebenen „als nötig“. Im zweidimensionalen kann man Entartung nur durch überflüssige Ungleichungen erzeugen. Im dreidimensionalen ist die Spitze einer Viereckspyramide entartet.



Proposition 3.6.2 *Ist x eine entartete Ecke, so hat x weniger als m Nichtnulleinträge.*

Beweis. x kann nur auf $B \cap B'$ von Null verschieden sein. □

Proposition 3.6.3 *a) Ein Pivot von der Basis B nach B' mit Pivotelement a_{ij} ist genau dann entartet, wenn $(A_{.B}^{-1}b)_i = 0 = (A_{.B'}^{-1}b)_i$ ist.*

b) Ist ein Pivot von Basis B zu B' nicht entartet, so ist

$$c^\top x' = c_{B'}^\top A_{.B'}^{-1} b > c_B^\top A_{.B}^{-1} b = c^\top x.$$

Beweis. Wir betrachten $A_{.B'}^{-1}b - A_{.B}^{-1}b = (\eta - I_n)A_{.B}^{-1}b$ mit einer η Matrix wie in Abschnitt 2.3 angegeben, die den Gauß-Jordan-Schritt beschreibt. Die Matrix $(\eta - I_n)$ hat genau eine von Null verschiedene Spalte, nämlich die i-te. Folglich ist $A_{.B'}^{-1}b = A_{.B}^{-1}b \leftrightarrow (A_{.B}^{-1}b)_i = 0$. Die zweite Behauptung folgt nun mit Lemma 3.4.7. □

Wir werden in den Übungen ein Beispiel kennenlernen, bei dem man unter Anwendung der steilste-Anstieg-Regel in einer degenerierten Ecke hängenbleibt und *zykelt*, d.h. es gibt eine Folge von Basen $B_1, \ldots, B_k = B_1$, so daß der Simplexalgorithmus unter Anwendung dieser Pivotregel von B_i nach B_{i+1} wechselt.

Kleine Rundungsfehler heben Degeneration auf. Es war lange Zeit Lehrmeinung, daß in der Praxis Zykeln nicht auftritt. In letzter Zeit wird häufen sich aber Berichte über Zykeln bei sehr großen, strukturierten Probleminstanzen.

Satz 3.6.4 *Bei Anwendung von Bland's rule zykelt das Simplexverfahren nicht.*

Beweis. Sei $B_1, \ldots, B_k = B_1$ ein Zykel und $B_{i+1} = B_i \setminus \{e_i\} \cup \{f_i\}$ und

$$J = \bigcup_{i=1}^{k} \{e_i\} \stackrel{!}{=} \bigcup_{i=1}^{k} \{f_i\}.$$

Sei $t = e_i = f_j$ der größte Index in J. Wenn t in die Basis B_j aufgenommen wird, müssen wegen Bland's rule die reduzierten Kosten aller Elemente in J nicht-positiv sein, speziell gilt dies für das Element $s := f_i$, welches in die Basis aufgenommen wird, wenn t die Basis verläßt. Wir haben also

$$\begin{aligned} c_s - c_{B_j}^\top A_{.B_j}^{-1} A_{.s} &\leq 0 \\ c_s - c_{B_i}^\top A_{.B_i}^{-1} A_{.s} &> 0 \end{aligned}$$

und somit

$$c_{B_i}^\top A_{.B_i}^{-1} A_{.s} - c_{B_j}^\top A_{.B_j}^{-1} A_{.s} < 0. \tag{3.1}$$

Bei der Wahl des basisverlassenden Elementes kommen jeweils die Elemente k mit $(A_{.B}^{-1} b)_k = 0$ in Betracht. Wenn t die Basis verläßt, muß also wegen Anwendung von Bland's rule $(A_{.B_i}^{-1} A_{.s})_k \leq 0$ für alle $k \in B_i \cap J \setminus \{t\}$ sein. Da ferner $B_i \setminus J \subseteq B_j$ ist erhalten wir insgesamt

$$\begin{aligned} c_{B_i}^\top A_{.B_i}^{-1} A_{.s} - c_{B_j}^\top A_{.B_j}^{-1} A_{.s} &= (c_{B_i}^\top - c_{B_j}^\top A_{.B_j}^{-1} A_{.B_i}) A_{.B_i}^{-1} A_{.s} \\ &= \underbrace{(c_t - c_{B_j}^\top A_{.B_j}^{-1} A_t)}_{>0} \underbrace{(A_{.B_i}^{-1} A_{.s})_t}_{>0} \\ &+ \underbrace{(c_{J \cap B_i \setminus \{t\}}^\top - c_{B_j}^\top A_{.B_j}^{-1} A_{.J \cap B_i \setminus \{t\}})}_{\leq 0} \underbrace{(A_{.B_i}^{-1} A_{.s})_{J \cap B_i \setminus \{t\}}}_{\leq 0} \\ &+ \underbrace{(c_{B_i \cap B_j}^\top - c_{B_j}^\top A_{.B_j}^{-1} A_{.B_i \cap B_j})}_{=0} (A_{.B_i} A_{.s})_{B_i \cap B_j} \\ &> 0 \end{aligned}$$

im Widerspruch zu (3.1). □

Damit können wir nun Endlichkeit und Korrektheit des Simplexverfahrens beweisen.

Satz 3.6.5 (Korrektheit des Simplexverfahrens) *a) Bei Anwendung von Bland's rule stoppt das Simplexverfahren nach endlich vielen Schritten.*

b) *Wenn das Simplexverfahren stoppt, und $c^\top - c_B^\top A_{.B}^{-1} A \not\leq 0$ ist, so ist das Problem unbeschränkt, andernfalls ist B eine optimale Basis.*

Beweis. Es gibt nur endlich viele Basen, nämlich $\binom{n}{m}$ viele. Wegen Satz 3.6.4 und Lemma 3.4.7 wird jede Basis höchstens einmal besucht. Der Rest folgt aus Proposition 3.4.4 und Lemma 3.4.9.

3.7 Bemerkungen zur Numerik

Es ist nicht-trivial, einen numerisch stabilen LP-Code zu schreiben. Es gibt im Netz eine lange Liste von Standardbeispielen, die gängige numerische Schwächen von Codes testen. Wir wollen hier mit ein paar Bemerkungen Probleme und Möglichkeiten der LP-Numerik anreißen.

Bei einer Implementierung wird man natürlich nicht das ganze Tableau abspeichern. Für die Basisspalten genügt es, ihre Nummern zu kennen, da sie nur die Einheitsmatrix enthalten (genauer sollte man sich die zugehörige Permutation merken). Das ergibt allerdings zusätzlichen Buchhaltungsaufwand, den wir uns hier sparen.

In der Praxis hat man es häufig mit dünnbesetzten Matrizen zu tun, das sind solche, bei denen die meisten Einträge Null sind. Hier setzt man (wie auch schon in der numerischen linearen Algebra) “sparse matrix” Techniken ein.

Üblicherweise wird man auch weder stets neu $A_{.B}^{-1}$ noch das Tableau berechnen. Beim Basiswechsel von B nach B' geht nach Abschnitt 2.3 $A_{.B'}^{-1}$ durch Multiplikation mit einer η-Matrix aus $A_{.B'}^{-1}$ hervor. Außerdem benötigt man zur Auswahl der nächsten Basis nur die reduzierten Kosten und die Daten der Pivotspalte. Also genügt es z.B., solange Einträge der reduzierten Kosten zu berechnen, bis man einen positiven Eintrag gefunden hat. Dann berechnet man die Einträge in der Pivotspalte zur Wahl der Pivotzeile, ändert die Basis und die Inverse. Es empfiehlt sich, wegen der Fehlerfortpflanzung bei der Matrixmultiplikation hin und wieder eine volle Matrixinversion durchzuführen.

3.8 Die Zweiphasenmethode

Wir wollen nun zu einem beliebigen linearen Programm in Standardform ein Hilfsproblem konstruieren, das die Voraussetzungen zur Anwendung des Simplexverfahrens erfüllt und dessen Optimallösung eine zulässige Basis des Ausgangsproblems bildet. Sei also

$$
\begin{array}{lll}
 & \max & c^\top x \\
(P) & \text{unter} & Ax = b \\
 & & x \geq 0
\end{array}
$$

ein lineares Programm in Standardform mit einer beliebigen Matrix $A \in \mathbb{R}^{m \times n}$. Das zugehörige Hilfsproblem ist dann

$$
\begin{array}{lll}
 & \max & -\sum_{i=1}^m y_i \\
(H) & \text{unter} & Ax + y = b \\
 & & x, y \geq 0
\end{array}
$$

mit Startbasis $B = \{n+1, \ldots, m+n\}$. Die Startbasis besteht also nur aus den *künstlichen Schlupfvariablen*, deren Summe wir minimieren wollen.

Proposition 3.8.1 *a) (A, I) hat vollen Rang und B ist zulässige Startbasis.*

b) Ist der optimale Zielfunktionswert von (H) ungleich 0, so ist (P) unzulässig.

c) Ist $(x, 0)$ eine optimale Basislösung zur Basis $B' \subseteq \{1, \ldots, n\}$ für (H), so ist x zulässig Basislösung zur Basis B' für (P) und A hat vollen Rang.

Beweis. Die erste Behauptung ist wegen $b \geq 0$ trivial. Wenn x zulässig für (P) ist, so ist $(x, 0)$ zulässig für (H). Da der Zielfunktionswert von $(x, 0)$ bzgl. (H) 0 ist und $y \geq 0$ gilt, folgt der Rest. □

Wenn man nun nach Lösen des Hilfsproblems (wir sprechen von Phase I) einen Zielfunktionswert 0 erreicht hat aber noch eine künstliche Schlupfvariable y_i in der Basis ist, so können wir diese gegen ein beliebiges Element x_k, $k \in \{1, \ldots, n\}$ mit $(A_{.B'}^{-1} A_{.k})_i \neq 0$ austauschen, denn mit $B'' := B' \setminus \{n+i\} \cup \{k\}$ haben wir, wenn wir $\tilde{A} := A_{.B'}^{-1} A$

$$A_{.B''}^{-1} = \eta A_{.B'}^{-1}$$

mit

$$
\eta = \begin{array}{cc}
 & \begin{array}{cccccc} & & & i & & \end{array} \\
 \begin{array}{c} \\ \\ \\ i \\ \\ \\ \end{array} &
\begin{pmatrix}
1 & 0 & \ldots & -\frac{\tilde{a}_{1k}}{\tilde{a}_{ik}} & \ldots & 0 \\
0 & 1 & \ldots & -\frac{\tilde{a}_{2k}}{\tilde{a}_{ik}} & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots & & \vdots \\
0 & 0 & \ldots & \frac{1}{\tilde{a}_{ik}} & \ldots & 0 \\
\vdots & \vdots & & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & -\frac{\tilde{a}_{mk}}{\tilde{a}_{ik}} & \ldots & 1
\end{pmatrix}
\end{array}.
$$

Nun ist $(A^{-1}_{.B'}b)_i = 0$, also $A^{-1}_{.B''}b = \eta A^{-1}_{.B'}b = A^{-1}_{.B'}b$.

Gibt es kein solches x_k, so ist wegen $(A^{-1}_{.B'})_{i.}(A, b) = (0, \ldots, 0)$ die Zeile redundant, kann also gestrichen werden. (Genauer, wenn y_j das i-te Basiselement ist, so ist $(A^{-1}_{.B'})_{i.}e_j = e_i$ also $(A^{-1}_{.B'})_{ij} = 1$, also ist die j-te Zeile von (A, b) eine Linearkombination der übrigen Zeile. Die Matrix hatte also nicht vollen Rang. Da nur Äquivalenzumformungen durchgeführt wurden, hat die transformierte Matrix, aus der die Nullzeile gestrichen wurde, den gleichen Rang, definiert also ein äquivalentes Gleichungssystem).

Iteriert man diese Vorgehensweise, so endet man mit einer regulären Matrix $\hat{A}$ und einer Basis, die keine künstliche Variable mehr enthält. Nun kann man die künstlichen Variablen mit ihren Spalten streichen und den Simplexalgorithmus bzgl. der Originalzielfunktion starten.

Wir erhalten also folgenden schematischen Ablauf:

Eingabedaten: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m, b \geq 0$, partielle zulässige Basis B (eventuell leer), $c \in \mathbb{R}^n$.

Phase 1a: Ergänze B durch künstliche Schlupfvariablen zu voller Basis, und minimiere die Summe der künstlichen Schlupfvariablen mit dem Simplexalgorithmus.

Phase 1b: Enthält die optimale Basis noch künstliche Variablen, pivotiere sie hinaus. Ist dies nicht möglich, streiche die zugehörige Zeile, Spalte und das Basiselement.

Phase 1c: Streiche alle Spalten künstlicher Schlupfvariablen.

Phase 2: Optimiere das Originalproblem mit dem Simplexverfahren.

Beispiel 3.8.2 *Wir betrachten folgendes Lineares Programm*

$$\begin{array}{rrcrcrcrcrcl}
\max & -3x_1 & + & 2x_2 & + & 2x_3 & - & 4x_4 & - & 2x_5 & & \\
\textit{unter} & -2x_1 & + & x_2 & + & x_3 & & & + & x_5 & = & 3 \\
& -2x_1 & + & x_2 & + & & & x_4 & & & = & 2 \\
& x_1 & + & x_2 & & & + & x_4 & & & = & 7
\end{array}$$

Als erste Basisvariable können wir x_3 oder x_5 wählen. Für die zweite und dritte Ungleichung brauchen wir künstliche Schlupfvariablen. *Das Hilfsproblem lautet dann:*

$$
\begin{array}{rl}
\max & -y_1 - y_2 \\
\textit{unter} & -2x_1 + x_2 + x_3 + x_5 = 3 \\
& -2x_1 + x_2 + x_4 + y_1 = 2 \\
& x_1 + x_2 + x_4 + y_2 = 7
\end{array}
$$

Wir erhalten also folgende Tableaus:

$$
\begin{array}{rrrrrrr|r}
0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\
\hline
-3 & 2 & 2 & -4 & -2 & 0 & 0 & 0 \\
\hline
-2 & 1 & 1 & 0 & 1 & 0 & 0 & 3 \\
-2 & 1 & 0 & 1 & 0 & 1 & 0 & 2 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 7
\end{array}
\qquad
\begin{array}{rrrrrrr|r}
-1 & 2 & 0 & 2 & 0 & 0 & 0 & 9 \\
\hline
-7 & 4 & 4 & -4 & 0 & 0 & 0 & 6 \\
\hline
-2 & 1 & 1 & 0 & 1 & 0 & 0 & 3 \\
-2 & \boxed{1} & 0 & 1 & 0 & 1 & 0 & 2 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 7
\end{array}
$$

$$
\begin{array}{rrrrrrr|r}
3 & 0 & 0 & 0 & 0 & -2 & 0 & 5 \\
\hline
1 & 0 & 4 & -8 & 0 & -4 & 0 & -2 \\
\hline
0 & 0 & 1 & -1 & 1 & -1 & 0 & 1 \\
-2 & 1 & 0 & 1 & 0 & 1 & 0 & 2 \\
\boxed{3} & 0 & 0 & 0 & 0 & -1 & 1 & 5
\end{array}
\qquad
\begin{array}{rrrrrrr|r}
0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\
\hline
0 & 0 & 4 & -8 & 0 & \frac{-11}{3} & \frac{-1}{3} & \frac{-11}{3} \\
\hline
0 & 0 & \boxed{1} & -1 & 1 & -1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & \frac{1}{3} & \frac{2}{3} & \frac{16}{3} \\
1 & 0 & 0 & 0 & 0 & \frac{-1}{3} & \frac{1}{3} & \frac{5}{3}
\end{array}
$$

Beim letzten Tableau ist die künstliche Zielfunktion Null und man hat mit $(\frac{5}{3}, \frac{16}{3}, 1, 0, 0)$ *eine zulässige Basislösung zur Basis* $B = \{1, 2, 3\}$ *gefunden. Nun kann man die künstliche Zielfunktion sowie die künstlichen Variablen streichen und erhält das optimale Tableau*

$$
\begin{array}{rrrrr|r}
0 & 0 & 0 & -4 & -4 & \frac{-23}{3} \\
\hline
0 & 0 & 1 & -1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & \frac{16}{3} \\
1 & 0 & 0 & 0 & 0 & \frac{5}{3}.
\end{array}
$$

Wir haben nun auch noch insgesamt folgenden Satz gezeigt:

Satz 3.8.3 *Wenn ein lineares Programm zulässig und beschränkt ist, so wird das Optimum an einer Ecke angenommen.*

Beweis. Ist das Programm zulässig, so findet man in Phase 1 eine zulässige Ecke. In Phase 2 terminiert der Simplexalgorithmus, da das Programm beschränkt ist in einer optimalen Ecke. □

3.9 Sensitivitätsanalyse

Anhand des optimalen Tableaus kann man verschieden Informationen über Änderungen der Lösung bei Änderung der Eingangsdaten herleiten. Wir wollen dies anhand zweier Beispiele diskutieren.

Beispiel 3.9.1 *Angenommen in dem Beispiel der Düngemittelfabrik (3.1.5) erfände ein Chemiker eine neue Formel für einen Dünger, der als Rohstoffvektor $A_{.3} = (3,2,2)^\top$ pro Tonne benötigt. Wie teuer muß das Unternehmen das Produkt absetzen können, damit die Produktion nach dieser Formel sich lohnt? Anstatt das lineare Programm von vorne zu lösen, können wir auch einfach ausnutzen, daß die reduzierten Kosten bzgl. der gegenwärtigen Basis $c_3 - c_B^\top A_{.B}^{-1} A_{.3}$ sind. Damit eine Aufnahme dieses Vektors in die Basis eine echte Verbesserung bringt, muß also*

$$\begin{aligned} c_3 &> (10,10,0)\begin{pmatrix} 3 \\ 2 \\ 2 \end{pmatrix} \\ &= 50 \end{aligned}$$

sein. Anhand der reduzierten Kosten kann man also Auswirkungen bei Einführung einer neuen Variablen studieren. Außerdem ist offensichtlich die Optimallösung bei Änderungen der Kostenfunktion in der Nichtbasis je sensibler, je geringer die reduzierten Kosten sind.

Im zweiten Beispiel wollen wir die Sensitivität gegenüber Änderungen der rechten Seite studieren. Offensichtlich bleibt, da sich die reduzierten nicht ändern, eine Basis optimal gegen Änderungen der rechten Seite um einen Vektor ε, solange $A_{.B}^{-1}(b+\varepsilon) \geq 0$ ist. Aus dieser Beziehung kann man für die rechte Seite obere und untere Schranken ausrechnen. Standardpakete für die lineare Programmierung bieten eine Sensitivitätsanalyse an.

Beispiel 3.9.2 *Betrachten wir das Beispiel der Ölraffinerie so liefert CPLEX aber sogar die Schranken, in denen sich die rechte Seite ändern darf, ohne daß der Zielfunktionswert sich wesentlich ändert.*

```
CPLEX> display sensitivity rhs
Display RHS sensitivity for which constraint(s): -
                                                RHS Sensitivity Ranges
Constraint Name      Dual Price         Down         Current            Up
RB1                     -2.8980    3457.4074       4000.0000     6344.1176
RB2                     -8.2560    3422.2222       5050.0000    27311.1111
RB3                     -7.1360    3437.5000       7100.0000    10810.1852
```

```
RB4                    -8.0640      1453.5714      4300.0000      7962.5000
O85                     0.4160    -14650.0000           zero     39850.0000
O90                     0.4293           zero           zero     25617.8571
O95                     0.4160    -14650.0000           zero     11385.7143
NF1                       zero      -infinity     15000.0000     16465.0000
NF2                       zero      3985.0000     10000.0000      +infinity
```

Setzen wir die rechte Seite von O85 allerdings nur auf .000018, so scheint sich die Lösung bereits zu ändern. Allerdings haben wir es hier offensichtlich mit Entartung zu tuen, so daß wir nicht auf eine Änderung der Basis schließen dürfen.

```
CPLEX> display problem all
Minimize
 ZIEL: - 9.97 X11 - 11.93 X12 - 14.13 X13 - 7.84 X21 - 9.8 X22 - 12 X23
 - 4.64 X31 - 6.6 X32 - 8.8 X33 - 2.24 X41 - 4.2 X42 - 6.4 X43
Subject To
 RB1: X11 + X12 + X13 <= 4000
 RB2: X21 + X22 + X23 <= 5050
 RB3: X31 + X32 + X33 <= 7100
 RB4: X41 + X42 + X43 <= 4300
 O85: - 17 X11 + X21 + 6 X31 + 14 X41 >= 0.000018
 O90: - 22 X12 - 4 X22 + X32 + 9 X42 >= 0
 O95: - 27 X13 - 9 X23 - 4 X33 + 4 X43 >= 0
 NF1: X11 + X21 + X31 + X41 >= 15000
 NF2: X13 + X23 + X33 + X43 <= 10000
Bounds
 All variables are >= 0.
CPLEX> optimize
Using devex.

Primal - Optimal:  Objective =   -1.3862559999e+05
Solution time =    0.00 sec.  Iterations = 0 (0)

CPLEX> display solution variables -
Variable Name           Solution Value
X11                        3485.806452
X13                         514.193548
X21                        5050.000000
X31                        7100.000000
X41                         829.193550
X43                        3470.806450
All other variables in the range 1-12 are zero.
```