LITERATUR

1. Werner, F.; Sotskov, Y.N.: Mathematics of Economics and Business, 1st Edition, Routledge, Abingdon (UK) / New York (USA), 2006, 536 p.

   KOSTENLOSER Download des Ebooks z.B. bei Amazon.de Kindle Store, siehe

   https://www.amazon.de/Mathematics-Economics-Business-English-Werner-ebook/dp/B000Q7ZFKW/

2. Neumann, K.; Morlock, M.: Operations Research, Carl Hanser Verlag, München, Wien, 1993.

3. Hillier, F; Lieberman, G: Introduction to Operations Research, 9th Edition, MacGraw Hill, New York (USA), 2010.

4. Taha, H.A.: Operations Research - An Introduction, 7th Edition, Prentice Hall, New York (USA), 2003.

5. **zur Auffrischung mathematischer Grundlagen:**
   Werner, F.: A Refresher Course in Mathematics, Bookboon Publishers, 2016, 284 S.

   KOSTENLOSER Download der pdf-Datei unter:

   https://bookboon.com/en/a-refresher-course-in-mathematics-ebook

1 Lineare Optimierung

1.1 Einführendes Beispiel

Beispiel 1: Futtermix

Ein Unternehmen produziert einen Futtermix bestehend aus drei Zutaten bezeichnet als R_1, R_2 und R_3. Die Zutaten R_1 und R_2 müssen mit einem Mindestprozentsatz enthalten sein, und Zutat R_3 darf einen Maximalprozentsatz nicht überschreiten. Die Preise sind bekannt, und die Daten sind wie in Tabelle 1:

Tabelle 1: Daten für Beispiel 1

Zutat	erforderlicher Prozentsatz	Preis in EUR per Kilogramm
R_1	mindestens 10 Prozent	25
R_2	mindestens 50 Prozent	17
R_3	höchstens 30 Prozent	12

Ziel: Bestimme einen zulässigen Futtermix mit minimalen Kosten

1.2 Voraussetzungen

Allgemeine Form eines LOP:

$$z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \longrightarrow \begin{cases} \max! \\ \min! \end{cases}$$

unter den Nebenbedingungen (u.d.N.)

$$\begin{array}{c} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \ \{\leq, =, \geq\} \ b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \ \{\leq, =, \geq\} \ b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \ \{\leq, =, \geq\} \ b_m \end{array}$$

$$x_j \geq 0 \ , j \in J \subseteq \{1, 2, \ldots, n\}$$

Alternativ kann man die folgende *Matrixdarstellung* eines LOP verwenden:

$$z = \mathbf{c}^T\mathbf{x} \longrightarrow \begin{cases} \max! \\ \min! \end{cases}$$

u.d.N.

$$\begin{array}{c} A\mathbf{x}\{\leq, =, \geq\}\mathbf{b} \\ \mathbf{x} \geq \mathbf{0}. \end{array} \tag{1}$$

Die Matrix A ist vom Format $m \times n$. $\mathbf{c} = (c_1, c_2, \ldots, c_n)^T$ ist der Vektor der Koeffizienten der Zielfunktion, und $\mathbf{b} = (b_1, b_2, \ldots, b_m)^T$ bezeichnet den Vektor der rechten Seite.

Geometrische Interpretation eines LOP mit zwei Variablen x_1 and x_2

Angenommen die Nebenbedingungen sind als Ungleichngen gegeben. Die Ungleichungen

$$a_{i1}x_1 + a_{i2}x_2 \, R_i \, b_i, \; R_i \in \{\leq, \geq\}, \; i = 1, 2, \ldots m,$$

sind Halbebenen begrenzt durch die Geraden

$$a_{i1}x_1 + a_{i2}x_2 = b_i.$$

Jede dieser Geraden kann in der Form

$$\frac{x_1}{s_1} + \frac{x_2}{s_2} = 1,$$

geschrieben werden, wobei $s_1 = b_i/a_{i1}$ und $s_2 = b_i/a_{i2}$ die Schnittpunkte der Geraden mit den Koordinatenachsen x_1 und x_2 sind.

Für konstantes z und $c_2 \neq 0$ ist die Zielfunktion

$$z = c_1 x_1 + c_2 x_2$$

eine Gerade der Form

$$x_2 = -\frac{c_1}{c_2} x_1 + \frac{z}{c_2},$$

d.h., für verschiedene Werte von z erhält man parallele Geraden mit dem Anstieg $-c_1/c_2$. Der Vektor $\mathbf{c} = (c_1, c_2)^T$ gibt die Richtung an, in die der Zielfunktionswert am stärksten steigt.

Also: Im Falle eines Maximierungsproblems mit der Zielfunktion z verschiebt man die Gerade

$$x_2 = -\frac{c_1}{c_2} x_1 + \frac{z}{c_2}$$

in die Richtung des Vektors $\mathbf{c}$, und bei der Minimierung von z wird die Gerade in the entgegengesetzte Richtung beschrieben durch den Vektor $-\mathbf{c}$ geschoben.

Ein LOP mit zwei Variablen kann *grafisch* wie folgt gelöst werden:

(1) Ermittle den zulässigen Bereich M (d.h. die Menge der zulässigen Lösungen als Durchschnitt aller zulässigen Halbebenen mit dem ersten Quadranten).

(2) Zeichne die Zielfunktion $z = Z$, wobei Z konstant ist und verschiebe sie parallel entweder in Richtung des Vektors $\mathbf{c}$ (im Fall $z \to \max!$) oder in die Richtung des Vektors $-\mathbf{c}$ (im Fall $z \to \min!$). Wende dieses Vorgehen an solange die Gerade $z = const$ gemeinsame Punkte mit dem zulässigen Bereich hat.

Definition 1: Eine zulässige Lösung $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, für die die Zielfunktion den optimalen (d.h. maximalen oder minimalen) Wert annimmt, heißt **optimale Lösung**.

1.3 Eigenschaften eines LOP

Definition 2: Eine Menge M heißt **konvex**, falls für je zwei beliebige Vektoren $\mathbf{x^1}, \mathbf{x^2} \in M$ jede Konvexkombination

$$\lambda \mathbf{x^1} + (1 - \lambda)\mathbf{x^2}$$

mit $0 \leq \lambda \leq 1$ auch zur Menge M gehört.

Definition 3: Ein Vektor (Punkt) $\mathbf{x} \in M$ heißt **Eckpunkt** der konvexen Menge M, falls $\mathbf{x}$ nicht als echte Konvexkombination von zwei anderen Vektoren aus M geschrieben werden kann, d.h. $\mathbf{x}$ kann nicht geschrieben werden als

$$\lambda \mathbf{x^1} + (1 - \lambda)\mathbf{x^2}$$

mit $\mathbf{x^1}, \mathbf{x^2} \in M$ und $0 < \lambda < 1$.

Theorem 1:
Die Menge M aller zulässigen Lösungen eines LOP ist entweder leer oder eine konvexe Menge mit einer endlichen Anzahl von Eckpunkten.

Theorem 2:
Falls die Menge M der zulässigen Lösungen eines LOP beschränkt ist, kann sie als Menge aller Konvexkombinationen der Eckpunkte $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^s}$ von M geschrieben werden, d.h.:

$$M = \Big\{ \mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} = \lambda_1 \mathbf{x^1} + \lambda_2 \mathbf{x^2} + \cdots + \lambda_s \mathbf{x^s};$$
$$0 \leq \lambda_i \leq 1, \quad i = 1, 2, \ldots, s, \quad \sum_{i=1}^{s} \lambda_i = 1 \Big\}$$

Sei M die Menge der zulässigen Lösungen und es werde die Maximierung der Funktion $z = \mathbf{c}^T\mathbf{x}$ betrachtet.

Dann gibt es die folgenden drei Fälle:

(a) Es gilt $M = \emptyset$. Dann widersprechen sich die Nebenbedingungen, d.h. es existiert keine zulässige Lösung des LOP.

(b) M ist eine nichtleere beschränkte Menge vom $\mathbb{R}^n$.

(c) M ist eine unbeschränkte Menge des $\mathbb{R}^n$, d.h. mindestens eine Variable kann beliebig groß werden, oder, falls die Variablen auch negativ sein können, kann eine von ihnen beliebig klein werden.

Im Fall (c) gibt es zwei Möglichkeiten:

(c1) Die Zielfunktion z ist von oben beschränkt. Dann existiert eine optimale Lösung des Maximierungsproblems.

(c2) Die Zielfunktion z ist nicht beschränkt von oben. Dann existiert keine (endliche) Optimallösung des betrachteten Maximierungsproblems.

Theorem 3:
Falls ein LOP eine (endliche) Optimallösung besitzt, dann existiert mindestens ein optimaler Eckpunkt, in dem die Zielfunktion den optimalen Wert annimmt.

Theorem 4:
Seien $P_1, P_2, \ldots, P_r$ beschrieben durch die Vektoren $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^r}$ optimale Eckpunkte. Dann ist jede Konvexkombination

$$\mathbf{x^0} = \lambda_1 \mathbf{x^1} + \lambda_2 \mathbf{x^2} + \ldots + \lambda_r \mathbf{x^r}, \quad \lambda_i \geq 0,$$

$$i = 1, 2, \ldots, r, \quad \sum_{i=1}^{r} \lambda_i = 1$$

auch optimal.

1.4 Standardform eines LOP

Definition 4: Ein System $A\mathbf{x} = \mathbf{b}$ von $p = r(A)$ linearen Gleichungen, wobei in jeder Gleichung eine Variable nur in dieser Gleichung auftritt und den Koeffizienten $+1$ hat, heißt lineares Gleichungssystem in **kanonischer Form**. Die eliminierten Variablen heißen **Basisvariable** (BV), während die anderen Variablen **Nichtbasisvariable** (NBV) genannt werden.

Folglich ist die Anzahl der Basisvariablen gleich dem Rang der Matrix A. Falls $r(A) = p < n$, kann das System in der Form

$$I\mathbf{x}^{\mathbf{B}} + A_N\mathbf{x}^{\mathbf{N}} = \mathbf{b},$$

geschrieben werden, wobei $\mathbf{x}^{\mathbf{B}}$ der p-dimensionale Vektor der Basisvariablen ist, $\mathbf{x}^{\mathbf{N}}$ ist der $(n-p)$-dimensionale Vektor der Nichtbasisvariablen, und A_N is eine Teilmatrix von A bestehend aus den Spaltenvektoren, die zu den NBV gehören.

Definition 5: Eine Lösung $\mathbf{x}$ des Gleichungssystems $A\mathbf{x} = \mathbf{b}$ in kanonischer Form, bei der alle Nichtbasisvariablen den Wert Null haben, heißt **Basislösung**.

Definition 6: Ein LOP der Form

$$z = \mathbf{c}^T\mathbf{x} \longrightarrow \max!$$

$$\text{u.d.N.} \qquad A\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0},$$

mit $A = (A_N, I)$ und $\mathbf{b} \geq \mathbf{0}$ heißt **Standardform** des LOP.

Die Standardform eines LOP ist durch die folgenden Eigenschaften gekennzeichnet:

- das LOP ist ein Maximierungsproblem;
- die Nebenbedingungen sind als Gleichungssystem in kanonischer Form mit nichtnegativen rechten Seiten gegeben und
- alle Variablen sind nichtnegativ.

Jedes LOP kann in die Standardform transformiert werden mittels der folgenden Regeln. Es werden alle möglichen Verletzungen der Standardform gemäß Definition 6 betrachtet.

(a) Eine Variable x_j is nicht notwendig nichtnegativ, d.h. x_j kann beliebige Werte annehmen. Dann wird die Variable x_j durch die Differenz zweier nichtnegativer Variablen ersetzt, d.h. wir setzen:

$$x_j = x_j^* - x_j^{**} \qquad \text{with} \quad x_j^* \geq 0 \quad \text{and} \quad x_j^{**} \geq 0.$$

Dann ergibt sich:

$$\begin{aligned} x_j^* > x_j^{**} &\iff x_j > 0 \\ x_j^* = x_j^{**} &\iff x_j = 0 \\ x_j^* < x_j^{**} &\iff x_j < 0. \end{aligned}$$

(b) Die Zielfunktion ist zu minimieren:

$$z \;=\; c_1x_1 \;+\; c_2x_2 \;+\; \cdots \;+\; c_nx_n \;\to\; \min!$$

Die Minimierung der Funktion z ist äquivalent zur Maximierung der Funktion $\bar{z} \;=\; -z$:

$$z \;=\; c_1x_1 + c_2x_2 + \cdots + c_nx_n \to \min! \iff$$

$$\bar{z} = -z \;= -c_1x_1 - c_2x_2 - \cdots - c_nx_n \to \max!$$

(c) Für eine rechte Seite gilt $b_i \;<\; 0$:

$$a_{i1}x_1 \;+\; a_{i2}x_2 \;+\; \cdots \;+\; a_{in}x_n \;=\; b_i \;<\; 0.$$

Multiplikation dieser Nebenbedingung mit -1 ergibt:

$$-a_{i1}x_1 \;-\; a_{i2}x_2 \;-\; \cdots \;-\; a_{in}x_n \;=\; -b_i \;>\; 0.$$

(d) Angenommen, einige Nebenbedingungen sind Ungleichungen:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i$$

oder

$$a_{k1}x_1 + a_{k2}x_2 + \cdots + a_{kn}x_n \geq b_k.$$

Dann führt man eine Schlupfvariable u_i bzw. eine Überschussvariable u_k ein,
und man erhält eine Gleichung:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + u_i = b_i \quad \text{mit} \quad u_i \geq 0$$

oder

$$a_{k1}x_1 + a_{k2}x_2 + \cdots + a_{kn}x_n - u_k = b_k \quad \text{mit} \quad u_k \geq 0\ .$$

(e) Angenommen, das gegebene System ist nicht in kanonischer Form, d.h. die Nebenbedingunen sind z.B. wie folgt gegeben:

$$\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &=& b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &=& b_2 \\ & \ldots & \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &=& b_m \end{array}$$

mit $\quad b_i \geq 0,\ i = 1, 2, \ldots, m;\ x_j \geq 0,\ j = 1, 2, \ldots, n.$

In der obigen Form enthält keine Nebenbedingung eine eliminierte Variable. Man führt in jeder Nebenbedingung eine künstliche Variable x_{Ai} als Basisvariable ein und erhält:

$$\begin{array}{llllllll} a_{11}x_1 + & a_{12}x_2 & + \cdots + & a_{1n}x_n + x_{A1} & & & = b_1 \\ a_{21}x_1 + & a_{22}x_2 & + \cdots + & a_{2n}x_n & + x_{A2} & & = b_2 \\ \multicolumn{7}{c}{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots} \\ a_{m1}x_1 + & a_{m2}x_2 & + \cdots + & a_{mn}x_n & & + x_{Am} & = b_m \end{array}$$

mit $\quad b_i \geq 0,\ i = 1, 2, \ldots, m;\ x_j \geq 0,\ j = 1, 2, \ldots, n,$

und $x_{Ai} \geq 0, i = 1, 2, \ldots, m.$

Beispiel 2:

Gegeben ist das folgende LOP:

$$
\begin{aligned}
& z = -x_1 + 3x_2 + x_4 \to \min! \\
\text{s.t.} \quad & x_1 - x_2 + 3x_3 - x_4 \geq 8 \\
& x_2 - 5x_3 + 2x_4 \leq -4 \\
& x_3 + x_4 \leq 3 \\
& x_2, x_3, x_4 \geq 0.
\end{aligned}
$$

1.5 Der Simplex-Algorithmus

Grundidee:

- Von einem Eckpunkt ausgehend (repräsentiert durch eine zulässige Baislösung gemäß einer Standardform des LOP), wird der Zielfunktionswert berechnet und geprüft, ob der Übergang zu einem benachbartem Eckpunkt (mittels eines Pivotschrittes) zu einer Verbesserung führt.

- Falls ja, führe diesen Schritt zum benachbarten Eckpunkt aus und überprüfe, ob weitere Verbesserungen mittels neuem Pivotschritt möglich sind. Sobald ein Eckpunkt erreicht wurde, von dem kein weiterer Pivotschritt zu einer Verbesserung führt, hat man eine Optimallösung gefunden.

Zur Anwendung dieses Vorgehens benötigt man ein Kriterium um zu entscheiden, ob ein Übergang zu einem benachbarten Eckpunkt den Zielfunktionswert verbessert.

Angenommen, der Rang der Matrix A ist gleich $m : r(A) = m$, d.h., in der kanonischen Form gibt es m BV unter den n Variablen, und die Anzahl der NBV ist folglich gleich $n' = n - m$.

Wir betrachten die *zulässige kanonische Form* mit den BV x_{Bi} und den NBV x_{Nj}:

$$x_{Bi} \;=\; \hat{b}_i \;-\; \sum_{j=1}^{n'} \hat{a}_{ij} x_{Nj} \;, \quad i = 1, 2, \ldots, m \quad (n' = n - m). \tag{2}$$

Dann kann die Zielfunktion z wie folgt geschrieben werden:

$$\begin{aligned} z \;&=\; c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \\ &=\; \underbrace{c_{B1} x_{B1} + c_{B2} x_{B2} + \ldots + c_{Bm} x_{Bm}}_{(BV)} \;+\; \underbrace{c_{N1} x_{N1} + c_{N2} x_{N2} + \ldots + c_{Nn'} x_{Nn'}}_{(NBV)} \\ &=\; \textstyle\sum_{i=1}^{m} c_{Bi} x_{Bi} \;+\; \sum_{j=1}^{n'} c_{Nj} x_{Nj}. \end{aligned}$$

Mit den Gleichungen (2) kann man die Basisvariablen ersetzen und die Zielfunktion nur in Abhängigkeit der Nichtbasisvariablen schreiben. Man erhält

$$z \;=\; \sum_{i=1}^{m} c_{Bi}\left(\hat{b}_i \;-\; \sum_{j=1}^{n'} \hat{a}_{ij} x_{Nj}\right) \;+\; \sum_{j=1}^{n'} c_{Nj} x_{Nj}$$

$$\;=\; \sum_{i=1}^{m} c_{Bi}\hat{b}_i \;-\; \sum_{j=1}^{n'} \left(\sum_{i=1}^{m} c_{Bi}\hat{a}_{ij} \;-\; c_{Nj}\right) x_{Nj}.$$

Außerdem definiert man:

$$z_0 \;=\; \sum_{i=1}^{m} c_{Bi}\hat{b}_i \quad \text{(Zielfunktionswert der Basislösung);} \tag{3}$$

$$g_j \;=\; \sum_{i=1}^{m} c_{Bi}\hat{a}_{ij} - c_{Nj} \qquad \text{(Koeffizient der NBV } \; x_{Nj} \text{ in der Zielfunktionszeile).} \tag{4}$$

Für die Berechnung von z_0 sei erinnert, dass in einer Basislösung alle Nichtbasisvariablen gleich Null sind.

Damit erhält man folgende Darstellung der Zielfunktion in Abhängigkeit von den Nichtbasisvariablen x_{Nj}:

$$z \;=\; z_0 \;-\; g_1 x_{N1} \;-\; g_2 x_{N2} \;- \ldots - \; g_{n'} x_{Nn'}.$$

Der Koeffizient g_j gibt die Veränderung des Zielfunktionswertes an, wenn die NBV x_{Nj} Basisvariable wird und ihr Wert um eine Einheit steigt. Jetzt ergibt sich das folgende Optimalitätskriterium.

Theorem 5: (Simplex-Kriterium)
Falls $g_j \geq 0,\ j = 1, 2, \ldots, n'$, für alle Koeffizienten der Nichtbasisvariablen in der Zielfunktionszeile gilt, ist die zugehörige Lösung optimal.

Folgerung: Falls eine Spalte l mit $g_l < 0$ in einer zulässigen Basislösung existiert, kann der Zielfunktionswert vergrößert werden durch Aufnahme dieses Spaltenvektors in die Menge der Basisvektoren und x_{Nl} wird Basisvariable im nächsten Schritt.

Ausgehend von dem Starttableau wenden wir die Kurzform des Simplextableaus an. Man verwendet eine zusätzliche Zeile für die Koeffizienten g_j zusammen mit dem Zielfunktionswert z_0.

	NBV	x_{N1}	x_{N2}	$\cdots$	$\mathbf{x_{Nl}}$	$\cdots$	$x_{Nn'}$		
BV	-1	c_{N1}	c_{N2}	$\cdots$	c_{Nl}	$\cdots$	$c_{Nn'}$	0	Q
x_{B1}	c_{B1}	$\hat{a}_{11}$	$\hat{a}_{12}$	$\cdots$	$\hat{a}_{1l}$	$\cdots$	$\hat{a}_{1n'}$	$\hat{b}_1$	
x_{B2}	c_{B2}	$\hat{a}_{21}$	$\hat{a}_{22}$	$\cdots$	$\hat{a}_{2l}$	$\cdots$	$\hat{a}_{2n'}$	$\hat{b}_2$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$	
$\mathbf{x_{Bk}}$	c_{Bk}	$\hat{a}_{k1}$	$\hat{a}_{k2}$	$\cdots$	$\mathbf{\hat{a}_{kl}}$	$\cdots$	$\hat{a}_{kn'}$	$\hat{b}_k$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$	
x_{Bm}	c_{Bm}	$\hat{a}_{m1}$	$\hat{a}_{m2}$	$\cdots$	$\hat{a}_{ml}$	$\cdots$	$\hat{a}_{mn'}$	$\hat{b}_m$	
z		g_1	g_2	$\cdots$	g_l	$\cdots$	$g_{n'}$	z_0	

Bestimmung der Pivotspalte l

Wähle eine Spalte $l, 1 \leq l \leq n'$, mit $g_l < 0$. Oft wird eine Spalte l gewählt mit

$$g_l \;=\; \min\{g_j \mid g_j < 0,\; j = 1, 2, \ldots, n'\}.$$

Bestimmung der Pivotzeile k

Nach einem Pivotschritt muss die resultierende Basislösung zulässig sein. Daher wählt man die Zeile k mit $1 \leq k \leq m$ und

$$\frac{\hat{b}_k}{\hat{a}_{kl}} = \min\left\{\frac{\hat{b}_i}{\hat{a}_{il}} \mid \hat{a}_{il} > 0,\; i = 1, 2, \ldots, m\right\}.$$

Zur Berechnung der obigen Quotienten wird eine Spalte Q am Ende des obigen Tableaus hinzugefügt, in die der Quotient eingetragen wird in jeder Zeile mit einem positiven Element in der gewählten Pivotspalte.

Wird Spalte l als Pivotspalte gewählt, wird die entsprechende Variable x_{Nl} zur Basisvariablen im nächsten Schritt. Mit Zeile k als Pivotzeile wird die entsprechende Variable x_{Bk} Nichtbasisvariable im nächsten Schritt. Das Element $\hat{a}_{kl}$ heißt *Pivotelement*.

Theorem 6:
Falls $g_l < 0$ für einen Koeffizienten einer Nichtbasisvariablen in der Zielfunktionszeile sowie $\hat{a}_{il} \leq 0$ für alle Koeffizienten in der Spalte l gilt, dann hat das LOP keine (endliche) Optimallösung.

Theorem 7:
Falls ein Koeffizient $g_l = 0$ in der Zielfunktionszeile einer optimalen Lösung existiert mit $\hat{a}_{il} > 0$ für mindestens einen Koeffizienten in Spalte l, dann existiert eine andere optimale Basislösung, in der x_{Nl} eine Basisvariable ist.

Simplex-Algorithmus

Schritt 1: Transformiere das LOP in die Standardform mit den Nebenbedingungen in kanonischer Form wie folgt (wir nehmen an, dass **keine** künstlichen Variablen für die Transformation notwendig sind):

$$A_N\mathbf{x_N} + I\mathbf{x_B} = \mathbf{b}, \qquad \mathbf{x_N} \geq \mathbf{0}, \qquad \mathbf{x_B} \geq \mathbf{0}, \qquad \mathbf{b} \geq \mathbf{0}.$$

Die Ausgangsbasislösung ist

$$\mathbf{x} = (\mathbf{x_N}, \mathbf{x_B})^T = (\mathbf{0}, \mathbf{b})^T$$

mit dem Zielfunktionswert

$$z_0 = \mathbf{c}^T\mathbf{x}.$$

Stelle das entsprechende Anfangstableau auf.

Schritt 2: Betrachte die Koeffizienten $g_j,\ j = 1, 2, \ldots, n'$, der Nichtbasisvariablen x_{Nj} in der Zielfunktionszeile.

Falls $g_j \geq 0$ für $j = 1, 2, \ldots, n'$, dann ist die gegenwärtige Basislösung optimal, STOP. Andernfalls gibt es einen Koeffizienten $g_j < 0$ in der Zielfunktionszeile.

Schritt 3: Bestimme eine Spalte l mit

$$g_l = \min\{g_j \mid g_j < 0,\ j = 1, 2, \ldots, n'\}$$

als Pivotspalte.

Schritt 4: If $a_{il} \leq 0$ für $i = 1, 2, \ldots, m$, STOP (in diesem Fall existiert keine Optimallösung für das Problem). Andernfalls gibt es

mindestens ein $a_{il} > 0$.

Schritt 5: Bestimme die Pivotzeile k so, dass

$$\frac{b_k}{a_{kl}} = \min\left\{\frac{b_i}{a_{il}} \mid a_{il} > 0,\ i = 1, 2, \ldots, m\right\}.$$

Schritt 6: Tausche die Basisvariable x_{Bk} in Zeile k mit der Nichtbasisvariablen x_{Nl} in Spalte l aus und berechne die folgenden Werte des neuen Tableaus:

$$\begin{aligned}
\hat{a}_{kl} &= \frac{1}{a_{kl}};\\
\hat{a}_{kj} &= \frac{a_{kj}}{a_{kl}}; \qquad \hat{b}_k = \frac{b_k}{a_{kl}}; \qquad j = 1, 2, \ldots, n',\ j \neq l;\\
\hat{a}_{il} &= -\frac{a_{il}}{a_{kl}}; \qquad i = 1, 2, \ldots, m,\ i \neq k;\\
\hat{a}_{ij} &= a_{ij} - \frac{a_{il}}{a_{kl}} \cdot a_{kj}; \qquad \hat{b}_i = b_i - \frac{a_{il}}{a_{kl}} \cdot b_k;\\
&\qquad i = 1, 2, \ldots, m,\ i \neq k; \quad j = 1, 2, \ldots, n',\ j \neq l.
\end{aligned}$$

Außerdem erhält man die folgenden Werte in der letzten Zeile des Tableaus:

$$\begin{aligned}
\hat{g}_l &= -\frac{g_l}{a_{kl}};\\
\hat{g}_j &= g_j - \frac{g_l}{a_{kl}} \cdot a_{kj}; \qquad j = 1, 2, \ldots, n',\ j \neq l;\\
\hat{z}_0 &= z_0 - \frac{g_l}{a_{kl}} \cdot b_k.
\end{aligned}$$

Beispiel 3:

Ein Unternehmen beabsichtigt drei Typen von Produktion P_1, P_2 und P_3 herzustellen, so dass die gesamten Produktionskosten 32 000 EUR nicht überschreiten. 420 Arbeitsstunden stehen zur Verfügung, und 30 Einheiten des Rohmaterials sind vorhanden. Außerdem sind die Daten gemäß Tabelle 2 gegeben.

Tabelle 2: Daten für das Beispiel 3

Produkt	P_1	P_2	P_3
Verkaufspreis (EUR/Stück)	1600	3000	5200
Produktionskosten (EUR/Stück)	1000	2000	4000
Erforderliches Rohmaterial per Stück	3	2	2
Arbeitszeit in Stunden per Stück	20	10	20

Das Ziel besteht in der Bestimmung der herzustellenden Mengen der Produkte, so dass der Gewinn maximal wird. Sei x_i die Anzahl der produzierten Stücke von $P_i,\ i \in \{1, 2, 3\}$.

Beispiel 4:

Betrachtet wird das folgende LOP:

$$\begin{aligned} & z = -2x_1 - 2x_2 \to \min! \\ \text{u.d.N.} \quad & x_1 - x_2 \geq -1 \\ & -x_1 + 2x_2 \leq 4 \\ & x_1, x_2 \geq 0. \end{aligned}$$

Beispiel 5:

Gegeben ist das folgende LOP:

$$\begin{aligned} & z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \to \min! \\ \text{u.d.N.} \quad & 2x_1 + x_2 + x_3 \geq 4000 \\ & x_2 + 2x_4 + x_5 \geq 5000 \\ & x_3 + 2x_5 + 3x_6 \geq 3000 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0. \end{aligned}$$

1.6 2-Phasen Simplexalgorithmus

Die Einführung künstlicher Variablen ist erforderlich, falls mindestens eine Nebenbedingung als Gleichung ohne eliminierte Variable vorliegt. Z.B. können die Nebenbedingungen folgende Gestalt haben:

$$\begin{array}{rl} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & = b_2 \\ \multicolumn{2}{c}{\dotfill} \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & = b_m \end{array}$$

$$x_j \geq 0,\ i = 1, 2, \ldots, n; \quad b_i \geq 0,\ i = 1, 2, \ldots, m.$$

Wie im Fall (e) in Abschnitt 1.4 diskutiert, führt man eine künstliche Variable x_{Ai} in jeder Gleichung ein. Zusätzlich ersetzt man die Ausgangs-Zielfunktion z durch eine Hilfszielfunktion z_I, die die Summe aller künstlichen Variablen minimiert (bzw. äquivalent die negative Summe aller künstlichen Variablen maximiert), denn das Ziel ist es, dass alle künstlichen Variablen den Wert Null annehmen, um die Zulässigkeit der Lösung zu garantieren.

Das ergibt folgendes LOP für Phase I:

$$z_I = -x_{A1} - x_{A2} - \cdots - x_{Am} \longrightarrow \max!$$

$$\begin{array}{lllllll} \text{u.d.N.} & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + x_{A1} & & & = b_1 \\ & a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & + x_{A2} & & = b_2 \\ \multicolumn{5}{c}{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots} \\ & a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & & + x_{Am} & = b_m \end{array} \tag{5}$$

$$x_j \geq 0,\ j = 1, 2, \ldots, n \quad \text{und} \quad x_{Ai} \geq 0,\ i = 1, 2, \ldots, m.$$

Angenommen man hat eine optimale Lösung des Hilfsproblems (5) mit der Simplexmethode gefunden, d.h. das Verfahren bricht ab mit $g_j \geq 0$ für alle Koeffizienten der Nichtbasisvariablen in der Zielfunktionszeile (bezüglich der Hilfsfunktion z_I).

Am Ende von Phase I sind die folgenden zwei (Haupt)fälle möglich:

(1) $z_I^{\max} < 0 \iff$ Das Ausgangsproblem hat keine zulässige Lösung.

(2) Es gilt $z_I^{\max} = 0$ und alle künstlichen Variablen sind Nichtbasisvariable. Dann repräsentiert diese Basislösung eine zulässige kanonische Form für das Ausgangsproblem, und man kann mit Phase II des Simplex-Algorithm gemäß Abschnitt 1.5 beginnen.

Beispiel 6:

Gegeben ist das folgende LOP:

$$\begin{aligned} z = x_1 - 2x_2 &\to \max! \\ \text{u.d.N.} \quad x_1 + x_2 &\leq 4 \\ 2x_1 - x_2 &\geq 1 \\ x_1, x_2 &\geq 0. \end{aligned}$$

Beispiel 7:

Wir wenden den 2-Phasen Simplexalgorithmus auf das Beispiel 1 an. Nach Transformation in die Standardform erhält man:

$$\begin{aligned} \overline{z} = -25x_1 - 17x_2 - 12x_3 &\to \max! \\ \text{u.d.N.} \quad x_1 + x_2 + x_3 + x_{A1} &= 100 \\ x_3 + x_4 &= 30 \\ x_1 + x_3 + x_5 &= 50 \\ x_2 + x_3 + x_6 &= 90 \\ x_1, x_2, x_3, x_4, x_5, x_6, x_{A1} &\geq 0. \end{aligned}$$

Beispiel 8:

Wir betrachten das folgende LOP:

$$
\begin{aligned}
z = x_1 + 2x_2 &\to \max! \\
\text{s.t.} \qquad x_1 - x_2 &\geq 1 \\
5x_1 - 2x_2 &\leq 3 \\
x_1, x_2 &\geq 0.
\end{aligned}
$$

2 Diskrete Optimierung

Grundbegriffe und Beispiele

Diskretes Optimierungsproblem:

$$f(\mathbf{x}) \to \min! \quad (\max!)$$
$$\mathbf{x} \in S$$

S ist eine diskrete Menge, d.h.

für alle $\mathbf{x} \in S$ existiert eine offene Umgebung, die außer $\mathbf{x}$ kein weiteres Element enthält (isolierte Punkte).

Spezialfall: S ist endlich.

$\to$ Oft wird S durch (lineare) Ungleichungen/Gleichungen beschrieben.

Ganzzahliges (lineares) Optimierungsproblem:

$$f(\mathbf{x}) = \mathbf{c}^T \cdot \mathbf{x} \to \min! \quad (\max!)$$

u.d.N.

$$A \cdot \mathbf{x} \leq \mathbf{b}$$
$$\mathbf{x} \in \mathbb{Z}_+^n$$

Parameter A, $\mathbf{b}$, $\mathbf{c}$ ganzzahlig

$\mathbb{Z}_+^n$ - Menge der ganzzahligen, nichtnegativen, n-dimensionalen Vektoren

Gemischt-ganzzahliges (lineares) Optimierungsproblem:

ersetze $\mathbf{x} \in \mathbb{Z}_+^n$ durch

$$x_1, x_2, \ldots, x_r \in \mathbb{Z}_+$$
$$x_{r+1}, x_{r+2}, \ldots, x_n \in \mathbb{R}_+$$

Binäres Optimierungsproblem:

ersetze $\mathbf{x} \in \mathbb{Z}_+^n$ durch

$$x_1, x_2, \ldots, x_n \in \{0, 1\}$$
$$\text{d.h.} \quad \mathbf{x} \in \{0, 1\}^n$$

Gemischt-binäres Optimierungsproblem:

ersetze $\mathbf{x} \in \mathbb{Z}_+^n$ durch

$$x_1, x_2, \ldots, x_r \in \{0, 1\}$$
$$x_{r+1}, x_{r+2}, \ldots, x_n \in \mathbb{R}_+$$

Kombinatorisches Optimierungsproblem (KOP):

Die Menge S ist endlich und nicht leer.

Anwendungen:

- Zuordnungsprobleme, z.B. Stundenplanprobleme, Frequenzzuweisung im Mobilfunk

- Reihenfolgeprobleme, z.B. Scheduling-Probleme, Travelling Salesman Problem (TSP)

- Auswahlprobleme, z.B. Rucksackproblem, Set Covering (Mengenüberdeckung), Set Partitioning (Mengenaufteilung)

Bemerkungen:

1. In der Regel lassen sich die Nebenbedingungen eines kombinatorischen Optimierungsproblems durch lineare Gleichungen/Ungleichungen mit ganzzahligen (oder binären) Entscheidungsvariablen $x_1, x_2, \ldots, x_n$ beschreiben.

   $\Rightarrow S$ ist Menge aller ganzzahligen Gitterpunkte eines konvexen Polyeders in $\mathbb{R}^n$.

2. Angenommen, Variable x kann nur endlich viele Werte $z_1, z_2, \ldots, z_k$ annehmen.

   $\Rightarrow$ Ersetze x durch k binäre Variablen $u_1, u_2, \ldots, u_k$ wie folgt:

$$x = z_1 \cdot u_1 + z_2 \cdot u_2 + \ldots + z_k \cdot u_k$$
$$u_1 + u_2 + \ldots + u_k = 1$$
$$u_i \in \{0, 1\}, \quad i = 1, 2, \ldots, k$$

3. Lineare Optimierungsprobleme sind in polynomialer Zeit lösbar. Jedoch sind Verfahren der linearen Optimierung (z.B. Simplexalgorithmus) nicht auf ganzzahlige und kombinatorische Probleme anwendbar.

Beispiel 1: Zuordnungsproblem

Beispiel 2: Investitionsplanung

Ein Unternehmen zieht 5 Projekte mit den folgenden Aufwendungen (in Mio. EUR) für die nächsten 3 Jahre in Betracht.

Projekt	Jahr 1	Jahr 2	Jahr 3	Gewinn
1	5	1	8	20
2	4	7	10	40
3	3	9	2	20
4	7	4	1	15
5	8	6	10	30
Verfügbare Mittel	25	25	25	

Welche Projekte sollten mit dem Ziel der Gewinnmaximierung realisiert werden?

Beispiel 3: Zuschnittoptimierung

Drei Leisten sind zur Herstellung eines bestimmten Erzeugnisses notwendig. Zwei Leisten müssen je 1,5 m und eine Leiste muss 2 m lang sein. Es stehen 300 Leisten mit einer Länge von je 6,5 m und 80 Leisten mit einer Länge von je 5,5 m zur Verfügung.

Wie sind die zur Verwendung stehenden Leisten zuzuschneiden, damit eine maximale Stückzahl des Erzeugnisses hergestellt werden kann? Es sollen alle Leisten genutzt werden.

Bestimmung aller Zuschnittvarianten:

(a) Eine 6,5 m Leiste kann nach folgenden Varianten in 2 m bzw. 1,5 m Leisten geteilt werden:

	Anzahl der 2 m Leisten	Anzahl der 1,5 m Leisten
Variante 1:	3	0
Variante 2:	2	1
Variante 3:	1	3
Variante 4:	0	4

(b) Eine 5,5 m Leiste kann nach folgenden Varianten in 2 m bzw. 1,5 m Leisten geteilt werden:

	Anzahl der 2 m Leisten	Anzahl der 1,5 m Leisten
Variante 5:	2	1
Variante 6:	1	2
Variante 7:	0	3

Modellierung verschiedener Sachverhalte:

1. Angenommen, bei der Serienproduktion eines Gutes treten Fixkosten β auf, wenn eine positive Menge produziert wird. Dabei beschreibt x die produzierten Mengeneinheiten.

$\rightarrow$ Kosten $c(x)$:

$$c(x) := \begin{cases} \alpha x + \beta & \text{falls } x > 0 \\ 0 & \text{falls } x = 0 \end{cases}$$

$\rightarrow$ Führe binäre Variable u ein mit

$$u := \begin{cases} 1 & \text{falls } x > 0 \\ 0 & \text{falls } x = 0 \end{cases}$$

$\Rightarrow$ Kosten haben die Form

$$c(x) = \alpha x + \beta u \quad (x \geq 0)$$

2. Mit Hilfe binärer Variablen lassen sich auch nicht zusammenhängende Bereiche durch Gleichungen bzw. Ungleichungen beschreiben.

Betrachte:

$$f(x_1, x_2) \to \min!$$

u.d.N.

$$(x_1, x_2) \in M_1 \cup M_2$$

$$\Rightarrow \quad f(x_1, x_2) \to \min!$$

u.d.N.

$$\begin{aligned} 2x_1 + 3x_2 &\leq 6 \\ x_1 + 2u_1 &\geq 2 \\ x_2 + u_2 &\geq 1 \\ u_1 + u_2 &\leq 1 \end{aligned}$$

$$x_1, x_2 \in \mathbb{R}_+$$
$$u_1, u_2 \in \{0, 1\}$$

$\Rightarrow$ für (u_1, u_2) sind 3 Paare zulässig:

$$\begin{aligned} (u_1, u_2) = (0, 0) \quad &\Rightarrow \nexists (x_1, x_2) \in M_1 \cup M_2 \\ (u_1, u_2) = (1, 0) \quad &\Rightarrow (x_1, x_2) \in M_1 \\ (u_1, u_2) = (0, 1) \quad &\Rightarrow (x_1, x_2) \in M_2 \end{aligned}$$

Grafische Lösung: Im Fall $n = 2$ lässt sich ein optimaler Gitterpunkt grafisch ermitteln.

2.2 Lösungsmethoden

2.1 Definitionen und Verfahrensklassen

exakte Verfahren: Bestimmung einer optimalen Lösung in endlich vielen Schritten

vollständige Enumeration: nur bei *sehr kleinen* Problemen möglich

Verfahrensklassen:

(a) *Branch and Bound Verfahren*

- Verfahren der impliziten Enumeration: Schließe sukzessiv Teilmengen von S aus, die keine optimale Lösung des Problems enthalten können.
- Grundidee für Minimierungsprobleme:
  - Verzweigung (Branch): Teile die Lösungsmenge in mindestens zwei (disjunkte) Teilmengen auf.
  - Beschränkung (Bound): Berechne für jede Teilmenge S_i eine untere Schranke LB_i (lower bound).
  - Sei UB eine bekannte obere Schranke (upper bound) und gilt $LB_i \geq UB$ für S_i, so braucht S_i nicht weiter betrachtet werden.

(b) *Schnittebenenverfahren (für ganzzahlige Optimierung)*

- Relaxiere die Lösungsmenge durch Streichung der Ganzzahligkeitsbedingung ($\rightarrow$ LOP).
- Bestimme die (stetige) Optimallösung $\mathbf{x}$ des LOP.
- Sind einzelne Komponenten von $\mathbf{x}$ nicht ganzzahlig, füge systematisch zusätzliche Nebenbedingungen (*Schnitte*) ein, wobei die ursprüngliche optimale Lösung abgeschnitten wird. Es darf kein ganzzahliger Punkt von S abgeschnitten werden.

(c) *Dynamische Optimierung* (siehe Kapitel 4)

(d) *Heuristische Verfahren*

- bestimmen nur eine Näherungslösung
- Konstruktive Verfahren: bestimmen eine zulässige Lösung (z.B. Greedy-Verfahren, Prioritätsregelverfahren)
- Iterative Verfahren: verbessern eine gegebene zulässige Lösung (z.B. Lokale Suche, Metaheuristiken - siehe Kapitel 3)
- Verkürzte exakte Verfahren: z.B. vorzeitig abgebrochene Branch and Bound Verfahren

Heuristische Verfahren

(a) Konstruktive Verfahren (Eröffnungsverfahren)

wichtige Klasse: Greedy-Algorithmen ('gieriger' Algorithmus)

- treffen in jedem Schritt eine 'lokal-optimale' Entscheidung
- führen in manchen Fällen auch zu einer global-optimalen Lösung

Beispiele und Anwendungen:

- Algorithmus von Kruskal zur Bestimmung eines Minimalgerüstes
- Algorithmus von Dijkstra zur Suche eines kürzesten Weges in einem ungerichteten Graphen
- Bestimmung einer Startlösung für das Transportproblem, z.B. mittels Zeilen- bzw. Spaltenminimumregel
- Einfügeverfahren für das Travelling Salesman Problem (TSP)

Bestimmung eines Minimalgerüsts

Definition 1: Ein zusammenhängender, alle Knoten eines gegebenen (ungerichteten) Graphen G enthaltender Teilgraph von G mit minimaler Kantenanzahl heißt **Gerüst** von G.

$$G = [V, E, c]$$

Bemerkungen:

1. G besitzt ein Gerüst. $\Leftrightarrow$ G ist zusammenhängend.
2. Ein Gerüst ist ein alle n Knoten enthaltender Baum mit $n - 1$ Kanten.

Definition 2: Ein Gerüst mit minimaler Summe der Kantenbewertungen heißt **Minimalgerüst**.

Algorithmus von Kruskal

Schritt 1:

Wähle eine Kante $[k, l]$ mit kleinster Bewertung c_{kl} aus. Diese bildet (mit den Knoten k, l) den Teilgraphen G^1.

Schritt i $(i = 2, 3, \ldots, n-1)$:

Füge dem bisher erhaltenen Teilgraphen G^{i-1} eine weitere Kante $[u, v]$ mit kleinstmöglicher Bewertung (sowie die Knoten u, v) hinzu, so dass der neue Teilgraph G^i (der nicht zusammenhängend sein muss) keinen Kreis enthält.

$\Rightarrow$ Algorithmus erzeugt eine Folge von Wäldern, wobei der Graph G^{n-1} genau $n-1$ Kanten besitzt und ein Baum (ungerichteter zusammenhängender kreisfreier Graph) ist.

Travelling Salesman Problem:

Ein Handelsvertreter möchte, beginnend in seinem Heimatort (Knoten 1), $n-1$ weitere Kundenorte (Knoten $2, \ldots, n$) in einer zu ermittelnden Reihenfolge besuchen und anschließend zum Ausgangspunkt zurückkehren. Dabei soll die insgesamt zurückgelegte Entfernung minimal sein.

Also: Im Graph ist ein Zyklus (Tour; Rundreise) mit minimaler Länge gesucht, der alle Knoten genau einmal enthält.

Konstruktionsverfahren für das TSP

Varianten:

– symmetrischer TSP

– asymmetrisches TSP

- **Heuristik des nächsten Nachbars***:*

  – Beginnend mit einem Startknoten wird als nächstes der verfügbare Knoten mit der kürzesten Entfernung gesucht und die Kante angefügt, bis eine vollständige Tour vorliegt.

- **Greedy-Heuristk***:*

  – Beginnend mit der kürzesten Kante werden schrittweise Kanten hinzugefügt, bis die Tour komplett ist. In jedem Schritt wird dabei die kürzestmögliche Kante gewählt, ohne die Zulässigkeit zu verletzen.

- **Einfügeverfahren***:*

  – Beginnend mit einer Tour bestehend aus einer Stadt werden Städte eingefügt, bis alle Städte besucht sind.

  Dabei existieren verschiedene Varianten, z.B.

  – **Nearest Insertion:** Einfügen der Stadt mit geringster Differenz zu einer Stadt aus der aktuellen Tour.

  – **Farthest Insertion:** Einfügen der Stadt, bei der die geringste Distanz zu einer Stadt aus der Tour maximal ist.

  – **Cheapest Insertion:** Einfügen der Stadt, die die geringste Zunahme der Tourlänge bewirkt.

  Einfügeposition: Füge die neue Stadt so ein, dass die Tourlängenzunahme minimial ist.

Verallgemeinerung: **Tourenoptimierung:**

Ausgehend von einem Depot 0 sollen n Kunden mit Fahrzeugen gleicher Ladakapazität Q mit einem Gut beliefert werden, wobei die Fahrzeiten von jedem Kunden zum Depot und vom Depot zu jedem Kunden bekannt sind.

Gesucht ist eine Menge von Fahrten mit minimaler Gesamtdauer, so dass:

- jede Fahrt am Depot beginnt und endet;

- jeder Kunde auf genau einer Fahrt bedient wird;

- die Fahrzeugkapazizät nicht überschritten wird sowie

- eine für jede Fahrt gleiche Maximaldauer nicht überschritten wird.

Savingsverfahren*:*

Schritt 1:

Bilde die Pendeltouren $(0, i, 0)$. für $i = 1, \ldots, n$

Berechne für alle $i, j = 1, \ldots, n$ mit $i \neq j$ die **Savings**

$$s_{ij} = t_{i0} + t_{0j} - t_{ij}$$

und speichere die positiven s_{ij} in einer Liste L.

Schritt 2: Falls $L = \emptyset$, STOP!

Andernfalls entferne das erste (d.h. größte) Element aus L, sei dies s_{ij}.

Ist s_{ij} ein zulässiges Saving, so verschmelze die entsprechenden beiden Touren mit der Verbindung (i, j).

Gehe zu Schritt 2.

(b) Iterative Verfahren (Verbesserungsverfahren)

→ siehe Kapitel 3

Software-Pakete zur Lösung ganzzahliger und binärer, linearer Optimierungsprobleme

(a) Excel Solver

(b) LINGO/LINDO

(c) MPL/CPLEX

Einige Bemerkungen zur Nutzung von (b) LINGO:

→ kann als Studentenversion heruntergeladen werden:

http://www.lindo.com

(maximal 150 Nebenbedingungen, 300 Variablen, 30 ganzzahlige Variablen)

Modellierung in LINGO

Entscheidungsvariablen

SETS:

WOCHE/1...10/: pmenge,pcost;

ENDSETS

→ definiert Vektoren pmenge und pcost der Dimension 10

Zielfunktion und Nebenbedingungen

MIN=@SUM(WOCHE(i): pcost(i)*pmenge(i));

→ Zielfunktion: $\sum_{i=1}^{10} \text{pcost}(i) \cdot \text{pmenge}(i)$ → min!

@FOR(WOCHE(i): pmenge(i)¡=30);

→ Nebenbedingung: $\text{pmenge}(i) \leq 30$ für alle $i = 1, \ldots, 10$

DATA:

pcost = 100,50,80,120,30,50,10,60,20,90;

ENDDATA

$\rightarrow$ Vektor pcost $= (100, 50, 80, 120, 30, 50, 10, 60, 20, 90)^T$ gesetzt

Generierung des Problems

LINGO $\rightarrow$ Generate $\rightarrow$ Display Model

Lösen des Problems

LINGO $\rightarrow$ Solve

Beispielmodellierung

Betrachtet wird das folgende ganzzahlige Problem der Produktionsplanung:

$$\text{z} = 6x_1 + 12x_2 + 10x_3 + 5x_4 \rightarrow \max!$$

$$\begin{array}{rrrrrrrrl} \text{u.d.N.} & 2x_1 & + & 2x_2 & + & 3x_3 & + & x_4 & \leq & 27 \\ & x_1 & + & 5x_2 & + & x_3 & + & 2x_4 & \leq & 25 \\ & x_1 & & & & & & & \geq & 7 \\ & & & & & x_3 & + & x_4 & = & 5 \end{array}$$

$$x_1, x_2, x_3, x_4 \in \mathbb{Z}_+$$

Es können vier Produkte in ganzzahligen Mengen hergestellt werden, $x_i \in \mathbb{Z}_+$ bezeichne die von Produkt i hergestellte Menge - im nachfolgenden Modell mit prod(i) bezeichnet. Der Gewinn soll maximiert werden und es bestehen die oben angegebenen Restriktionen.

LINGO-Modell:

```
MODEL:
SETS:
PRODUKT/1..4/: prod, gewinn;
ENDSETS
MAX = @SUM(PRODUKT(i): gewinn(i)*prod(i));
2*prod(1) + 2*prod(2) + 3*prod(3) + prod(4) <= 27;
prod(1) + 5*prod(2) + prod(3) + 2*prod(4) <= 25;
prod(1) >= 7;
prod(3) + prod(4) = 5;
@FOR(PRODUKT(i): @GIN(prod(i)));
   DATA:
gewinn = 6, 12, 10, 5;
ENDDATA
END
```

Optimallösung:

$x_1 = 7, \quad x_2 = 2; \quad x_3 = 2; \quad x_4 = 3;$

$z = 101$

Bemerkungen

- Zwei Dateitypen in LINGO: .lng (reine Textdateien) und .lg4
- Variablen standardmäßig rational, nichtnegativ

  @GIN ganzzahlig, nichtnegative Variable

  @BIN binäre Variablen

  @FREE frei, d.h. nicht vorzeichenbeschränkt

  z.B. @FOR(WOCHE(i):@GIN(pmenge(i)));

  $\Rightarrow$ pmenge(i) nichtnegativ und ganzzahlig
- logische Operationen: #EQ#, #NE#, #GE#, #GT#, #LE#, #LT#, #AND#, #OR#
- LINGO nicht 'case-sensitive': SUM, sum, SuM identisch

Das Rucksackproblem

Problem: *Ein Bergsteiger hat n Gegenstände* $1, 2, \ldots, n$ *zur Verfügung, wobei*

c_i - Wert von Gegenstand i

a_i - Volumen von Gegenstand i

V - Volumen des Rucksacks

Ziel: *Bestimme eine Rucksackfüllung mit maximalem Gesamtwert, wobei das Volumen V nicht überschritten wird.*

$\Rightarrow$ Führe eine binäre Variable x_i ein wie folgt:

$$x_i = \begin{cases} 1, \text{ falls Gegenstand } i \text{ in den Rucksack gepackt wird} \\ 0, \text{ sonst} \end{cases}$$

$$\text{für } i = 1, 2, \ldots, n$$

$\Rightarrow$ mathematisches Modell:

$$\sum_{i=1}^{n} c_i x_i \to \max!$$

u.d.N.

$$\left.\begin{array}{l} \displaystyle\sum_{i=1}^{n} a_i x_i \leq V \\ x_1, x_2, \ldots, x_n \in \{0,1\} \end{array}\right\} =: S \qquad \text{(R)}$$

Das Problem (R) ist ein binäres Optimierungsproblem mit nur einer Nebenbedingung.

Greedy-Algorithmus:

Schritt 1:

Nummeriere die n Gegenstände nach nichtwachsenden Quotienten $\frac{c_i}{a_i}$ und setze $f_G := 0$.

Schritt 2:

Führe für $j = 1, 2, \ldots, n$ aus:

Falls $a_j > V$, setze $k := j$, $x_j^G := 0$ und gehe zu Schritt 3, andernfalls setze $x_j^G := 1$, $f_G := f_G + c_j$ und $V := V - a_j$.

Schritt 3:

Führe für $j = k+1, k+2, \ldots, n$ aus:

Falls $a_j > V$, setze $x_j^G := 0$, andernfalls setze $x_j^G := 1$, $f_G := f_G + c_j$ und $V := V - a_j$.

k - kritischer Index

Der Greedy-Algorithmus für das ganzzahlige Rucksackproblem

Ersetze im binären Problem $\mathbf{x} \in \{0,1\}^n$ durch $\mathbf{x} \in \mathbb{Z}_+^n$.

Schritt 1:

Nummeriere die Gegenstände nach nichtwachsenden Quotienten $\frac{c_i}{a_i}$ und setze $f_G := 0$.

Schritt 2:

Führe für $j = 1, 2, \ldots, n$ aus:

setze $x_j^G := \left[\frac{V}{a_j}\right]$; $\qquad f_G := f_G + c_j x_j^G$

und $V := V - a_j x_j^G$.

Bemerkung: Sei f^* der optimale Zielfunktionswert, dann gilt:

$$f^* \leq c_1 \frac{V}{a_1}\ , \quad f_G \geq c_1 \left[\frac{V}{a_1}\right]$$

$$f^* - f_G \leq c_1 \left(\frac{V}{a_1} - \left[\frac{V}{a_1}\right]\right) < c_1 \leq \max_{j=1,\ldots,n} c_j$$

Interpretation: c_j klein $\Rightarrow$ häufig $\mathbf{x}^G$ gut

3 Metaheuristiken

3.1 Iterative Verfahren, Grundbegriffe

Lokale Suche (Nachbarschaftssuche)

Führe eine Nachbarschaftsstruktur wie folgt ein:

$$N \ : \ S \to 2^S$$

$$\mathbf{x} \in S \ \Rightarrow \ N(\mathbf{x}) \subseteq 2^S$$

S - Menge der zulässigen Lösungen

$N(\mathbf{x})$ - Menge der Nachbarn der zulässigen Lösung $\mathbf{x} \in S$

Basisalgorithmus: Iterative Verbesserung
(Minimierung)

1. Bestimme eine Startlösung $\mathbf{x} \in S$;

   REPEAT

2. Ermittle die beste Lösung $\mathbf{x}' \in N(\mathbf{x})$;
3. IF $f(\mathbf{x}') < f(\mathbf{x})$ THEN $\mathbf{x} := \mathbf{x}'$;
4. UNTIL $f(\mathbf{x}') \geq f(\mathbf{x})$.

$\mathbf{x}'$ - lokaler Minimalpunkt bzgl. Nachbarschaft N

$\rightarrow$ Der Algorithmus arbeitet nach dem Prinzip der 'größten Verbesserung' *(best-fit)*.

Modifikation:

Wende das Prinzip der 'ersten Verbesserung' *(first-fit)* an, d.h. durchsuche Nachbarn in systematischer Weise und akzeptiere Nachbarn mit besserem Zielfunktionswert (Zfw.) sofort als Startlösung für die nächste Iteration.

(Abbruch folgt, wenn ein voller Zyklus aller Nachbarn ohne eine Zielfunktionswertverbesserung durchlaufen wurde.)

| **N(x)** | **sehr groß** $\Rightarrow$ Erzeuge Nachbarn zufällig.

$\Rightarrow$ Ersetze Zeile 2 im Algorithmus 'Iterative Verbesserung' durch

$$2^*\text{: Ermittle eine Lösung } \mathbf{x'} \in N(\mathbf{x})$$

Abbruch, falls

- vorgegebene maximale Rechenzeit verbraucht ist oder
- vorgegebene Anzahl zulässiger Lösungen erzeugt wurde oder
- vorgegebene Anzahl von Lösungen seit letzter Zielfunktionswertverbesserung erzeugt wurde, ohne den Zielfunktionswert weiter zu verbessern.

Betrachtet wird

$$\boxed{\begin{array}{l} f(\mathbf{x}) \rightarrow \min! \quad (\max!) \\ \text{u.d.N.} \\ \quad \mathbf{x} \in S \subseteq \{0,1\}^n \end{array}} \qquad (3.1)$$

Nachbarschaft $N_k(\mathbf{x})$:

$$N_k(\mathbf{x}) = \{\mathbf{x}' \in S \mid \sum_{i=1}^{n} \mid x_i - x_i' \mid \leq k\}$$

$$\Rightarrow \mid N_1(\mathbf{x}) \mid \quad \leq n$$

$$\mid N_2(\mathbf{x}) \mid \quad \leq n + \binom{n}{2} = n + \frac{n(n-1)}{2} = \frac{n(n+1)}{2}$$

Also:

$\mathbf{x}' \in N_k(\mathbf{x}') \iff$

$\mathbf{x}'$ zulässig und unterscheidet sich höchstens in k Komponenten von $\mathbf{x}$

Zur systematischen Erzeugung der Nachbarn ändere Komponente 1,2,... usw.

Minimalgerüst mit Nebenbedingungen

Angenommen, Kante $[u, v]$ kann nur zum Gerüst gehören, falls eine Kante $[k, l]$ zum Gerüst gehört oder eine von mehreren Kanten muss zum Gerüst gehören.

$\rightarrow$ Greedy-Algorithmus liefert nicht notwendig eine optimale Lösung.

Sei G ein Gerüst

$\Rightarrow N(G)$ - Menge aller Gerüste G^*, die sich von G durch genau eine Kante unterscheiden.

iterative Verbesserung:

Verfahren brechen in einem lokalen Optimum ab.

Metaheuristiken:

- Prinzipien zur Steuerung heuristischer Verfahren
- bauen auf lokaler Suche auf
- erlauben Übergänge zu Lösungen mit schlechterem Zfw.

3.2 Simulated Annealing

Randomisiertes Verfahren, da:

- $\mathbf{x}' \in N(\mathbf{x})$ wird zufällig gewählt
- im i-ten Schritt wird $\mathbf{x}'$ mit der Wahrscheinlichkeit

$$\min\left\{1,\ e^{\left(-\frac{f(\mathbf{x}')-f(\mathbf{x})}{t_i}\right)}\right\}$$

als neue Startlösung akzeptiert.

(t_i) - Folge positiver Steuerparameter mit $\lim\limits_{i\to\infty} t_i = 0$

('Temperatur')

Interpretation: (Minimierung!)

$f(\mathbf{x}') < f(\mathbf{x}) \ \Rightarrow$ Verbesserung wird immer akzeptiert

$f(\mathbf{x}') \geq f(\mathbf{x}) \ \Rightarrow$

t_i fixiert: $f(\mathbf{x}') - f(\mathbf{x})$ groß, dann wird $\mathbf{x}'$ mit kleiner Wahrscheinlichkeit akzeptiert

$f(\mathbf{x}') - f(\mathbf{x})$ fixiert: t_i klein, dann wird $\mathbf{x}'$ mit kleiner Wahrscheinlichkeit akzeptiert

Algorithmus Simulated Annealing

1. $i := 0$; wähle t_0;
2. bestimme eine Startlösung $\mathbf{x} \in S$;
3. $best := f(\mathbf{x})$;
4. $\mathbf{x}^* := \mathbf{x}$;

   REPEAT
5. erzeuge zufällig eine Lösung $\mathbf{x}' \in N(\mathbf{x})$;
6. IF $rand[0,1] < \min \left\{ 1, e^{\left(-\frac{f(\mathbf{x}')-f(\mathbf{x})}{t_i}\right)} \right\}$ THEN $\mathbf{x} := \mathbf{x}'$;
7. IF $f(\mathbf{x}') < best$ THEN

   BEGIN $\mathbf{x}^* := \mathbf{x}'$; $best := f(\mathbf{x}')$ END;
8. $t_{i+1} := g(t_i)$;
9. $i := i + 1$;

   UNTIL Abbruchkriterium ist erfüllt

Fragen

1. Wahl von g: $t_{i+1} := \alpha \cdot t_i$ mit $0 < \alpha < 1$ (geometrisches Abkühlungsschema)

   (oft: Zyklus mit konstanter Temperatur, danach Abkühlung)

2. Abbruchkriterium: analog zu lokaler Suche

Modifikation: Threshold Accepting

deterministisches Akzeptanzkriterium: akzeptiere $\mathbf{x}' \in N(\mathbf{x})$ falls $f(\mathbf{x}') - f(\mathbf{x}) \leq t_i$

$t_i \geq 0$ - Threshold im i-ten Schritt

3.3 Tabu-Suche

Grundstrategie:

Speichere bereits überprüfte Lösungen in einer Tabu-Liste TL und akzeptiere nur Nachbarn, die nicht in TL enthalten sind.

$$\rightarrow \text{aufwändig}$$

- Nutze *Attribute*, um zuletzt besuchte Lösungen zu charakterisieren.
- Länge der Tabu-Liste:

  konstant: Enthält die Liste t Attribute, so lösche bei jedem Übergang das älteste Attribut und nimm ein neues Attribut auf.

  variabel:

  - Lösche die TL bei Verbesserung des besten Zfw.
  - Nutze Schranken: $L_{min} \leq \ \mid TL \mid \ \leq L_{max}$.
  - Setze $\mid TL \mid = \mid TL \mid +1$ falls $f(\mathbf{x}') > f(\mathbf{x})$
    oder $\mid TL \mid = \mid TL \mid -1$ falls $f(\mathbf{x}') < f(\mathbf{x})$

- Nutzung des *Aspiration-Kriteriums*:

  verbotene Lösungen werden trotzdem akzeptiert (z.B. bei Verbesserung des besten Zfw.)

- $Cand(\mathbf{x}) = \{\mathbf{x}' \in N(\mathbf{x}) \mid$ Übergang von $\mathbf{x}$ zu $\mathbf{x}'$ ist nicht tabu oder $\mathbf{x}'$ erfüllt das Aspiration-Kriterium $\}$ ('erlaubte Übergänge')

  $\mid N(\mathbf{x}) \mid$ klein $\Rightarrow$ Wähle besten Nachbarn $\mathbf{x}'$ aus $Cand(\mathbf{x})$.

  $\mid N(\mathbf{x}) \mid$ groß $\Rightarrow$ Untersuche nur eine Teilmenge $V \subset Cand(\mathbf{x})$ und wähle besten Nachbarn $\mathbf{x}' \in V$.

Algorithmus Tabu-Suche

1. bestimme eine Startlösung $\mathbf{x} \in S$;
2. $best := f(\mathbf{x})$;
3. $\mathbf{x}^* := \mathbf{x}$;
4. $TL := \emptyset$;

   REPEAT
5. bestimme $Cand(\mathbf{x}) = \{\, \mathbf{x}' \in N(\mathbf{x}) \mid$ der Übergang von $\mathbf{x}$ zu $\mathbf{x}'$ ist nicht tabu oder $\mathbf{x}'$ erfüllt das Aspiration-Kriterium $\}$;
6. wähle eine Lösung $\overline{\mathbf{x}} \in Cand(\mathbf{x})$;
7. aktualisiere TL;
8. $\mathbf{x} := \overline{\mathbf{x}}$;
9. IF $f(\overline{\mathbf{x}}) < best$ THEN

   BEGIN $\mathbf{x}^* := \overline{\mathbf{x}}$; $best := f(\overline{\mathbf{x}})$ END;

   UNTIL Abbruchkriterium ist erfüllt

Erweiterung: Diversifikation

'Gute Lösungen' werden gespeichert. Wird der beste Zfw. während einer vorgegebenen Anzahl von Iterationen nicht verbessert, springe zu einer 'guten Lösung' zurück. (LONG-TERM MEMORY)

3.4 Genetische Algorithmen

- Nutzung von Darwins Evolutionstheorie ('survival of the fittest')
- arbeitet mit einer *Population* von Individuen (*Chromosomen*), die durch ihre Fitness charakterisiert werden, z.B.

  fitness(ch)=$f(\mathbf{x})$ bei $f \to \max!$

  fitness(ch)=$\frac{1}{f(\mathbf{x})}$ bei $f \to \min!$ und $f(\mathbf{x}) > 0$,
  wobei ch die Codierung der Lösung $\mathbf{x} \in S$ ist

$$\mathbf{x} = (0,1,1,1,0,1,0,1)^T \in \{0,1\}^8$$

ch: |0|1|1|1|0|1|0|1|

Erzeugung von Nachkommen mittels genetischer Operatoren

Mutation:

'Mutiere' Gene eines Individuums.

Elternchromosom | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

(3,5)-Inversion | 0 | 1 | **0** | **1** | **1** | 1 | 0 | 1 |

2-Mutation | 0 | **0** | 1 | 1 | 0 | 1 | 0 | 1 |

(1,4,7)-Mutation | **1** | 1 | 1 | **0** | 0 | 1 | **1** | 1 |

Crossover:

Kombiniere die genetischen Strukturen zweier Individuen und bilde zwei Nachkommen.

1-Punkt-Crossover z.B. (4,8)-Crossover

E_1 1 0 1 | 0 0 1 0 1 N_1 1 0 1 | **1 0 0 1 1**

$\rightarrow$

E_2 0 1 1 | 1 0 0 1 1 N_2 0 1 1 | **0 0 1 0 1**

2-Punkt-Crossover z.B. (3,5)-Crossover

E_1 1 0 | 1 0 0 | 1 0 1 N_1 1 0 | **1 1 0** | 1 0 1

$\rightarrow$

E_2 0 1 | 1 1 0 | 0 1 1 N_2 0 1 | **1 0 0** | 0 1 1

$\Rightarrow$ Bilde eine neue Generation mit Hilfe der Fitness-Werte entweder nur aus Nachkommen oder aus Eltern und Nachkommen (z.B. gemäß der Fitness der Individuen).

Die Auswahl der Eltern ist häufig proportional zur Fitness (roulette wheel selection).

Algorithmus GEN-ALG

1. Setze Parameter Populationsgröße $POPSIZE$, maximale Generationenanzahl
   $MAXGEN$, Wkt. P_{CO} für die Anwendung von Crossover und Wkt. P_{MU} für die Anwendung einer Mutation;
2. Erzeuge die Startpopulation POP_0 mit $POPSIZE$ Individuen (Chromosomen);
3. Bestimme die Fitness aller Individuen;
4. $k := 0$;

   WHILE $k < MAXGEN$ DO

   BEGIN
5. $h := 0$;

   WHILE $h < POPSIZE$ DO

   BEGIN
6. Wähle zwei Eltern aus POP_k proportional ihrer Fitness-Werte zufällig aus;
7. Wende mit Wkt. P_{CO} auf die ausgewählten Eltern ein Crossover an;
8. Wende mit Wkt. P_{MU} auf die entstandenen Individuen eine Mutation an;
9. $h := h + 2$;

   END
10. $k := k + 1$;
11. Wähle aus den erzeugten Nachkommen (bzw. auch den Eltern) $POPSIZE$ Individuen der k-ten Generation POP_k aus (z.B. proportional ihrer Fitness-Werte);

    END

Weitere Metaheuristiken:

- Ameisenalgorithmen (ant colony optimization): populationsorientiert, modellieren Verhalten von Ameisen auf der Wegsuche
- Partikelschwarmoptimierung: populationsorientiert, imitieren Verhalten von Vogel- oder Fischschwärmen
- Variable Nachbarschaftssuche: nutzt systematisch eine Reihe von Nachbarschaften, oft: $N^1 \subseteq N^2 \subseteq \ldots \subseteq N^k$

4 Dynamische Optimierung

$\rightarrow$ Es werden Probleme betrachtet, die in einzelne 'Stufen' zerlegt werden können, so dass die Gesamtoptimierung durch eine 'stufenweise Optimierung' ersetzt werden kann.

$\rightarrow$ Häufig wird sie angewendet bei der optimalen Steuerung wirtschaftlicher Prozesse, wobei die Stufen einzelnen Zeitperioden entsprechen.

4.1 Einführungsbeispiele

(a) Das Lagerhaltungsproblem

Problemstellung:

- Ein Gut werde während eines endlichen Planungszeitraumes, der aus n Perioden besteht, gelagert.

- In jeder Periode werde das Lager zu Beginn beliefert.
- In jeder Periode trete Nachfrage auf, die unmittelbar nach Belieferung in dieser Periode zu befriedigen ist.

Bezeichnungen:

$u_j \geq 0$ - die zu Beginn der Periode j gelieferte Menge

$r_j \geq 0$ - Nachfrage in Periode j

x_j - Lagerbestand unmittelbar vor Belieferung des Lagers in Periode j $(j = 1, 2, \ldots, n)$

Optimierungsproblem:

$$\sum_{j=1}^{n} \left[K\delta(u_j) + hx_{j+1} \right] \rightarrow \min!$$

u.d.N.

$$\begin{aligned} x_{j+1} &= x_j + u_j - r_j, && j = 1, 2, \ldots, n \\ x_1 &= x_{n+1} = 0 \\ x_j &\geq 0, && j = 2, 3, \ldots, n \\ u_j &\geq 0, && j = 1, 2, \ldots, n \end{aligned}$$

(4.1)

Bemerkung:

$$x_1 = x_{n+1} = 0 \text{ und } (4.1)$$

$\Rightarrow$ Ersetze in der Zielfunktion hx_{j+1} durch hx_j, damit jeder Summand die Form $g_j(x_j, u_j)$ hat.

$$x_j = x_{j+1} - u_j + r_j \geq 0 \quad \Rightarrow \quad u_j \leq x_{j+1} + r_j$$

Nebenbedingungen lassen sich wie folgt formulieren:

$$
\begin{aligned}
x_1 &= x_{n+1} = 0 \\
x_j &= x_{j+1} - u_j + r_j, \quad && j = 1, 2, \ldots, n \\
x_j &\geq 0, && j = 1, 2, \ldots, n \\
0 &\leq u_j \leq x_{j+1} + r_j, && j = 1, 2, \ldots, n
\end{aligned}
$$

(b) Das binäre Rucksackproblem

$$u_j := \begin{cases} 1, & \text{falls Gegenstand } j \text{ in den Rucksack gepackt wird} \\ 0, & \text{sonst} \end{cases}$$

Optimierungsproblem:

$$\sum_{j=1}^{n} c_j u_j \ \rightarrow \max!$$

u.d.N.

$$\sum_{j=1}^{n} a_j u_j \qquad \leq V$$

$$u_1, u_2, \ldots, u_n \in \{0, 1\}$$

$\rightarrow$ Hier sind die Stufen keine Zeitperioden, sondern es erfolgt eine 'künstliche Dynamisierung'.

Die Entscheidungen, welche der Gegenstände

$$1, 2, \ldots, n$$

in den Rucksack gepackt werden, werden als Entscheidungen in n aufeinanderfolgenden Stufen interpretiert.

x_j - Restvolumen des Rucksacks für die Gegenstände $j,\ j+1, \ldots, n$

$\Rightarrow x_1 = V \quad \text{und} \quad x_{j+1} = x_j - a_j u_j \quad \text{für alle } j = 1, 2, \ldots, n$

Umformuliertes Optimierungsproblem:

$$\sum_{j=1}^{n} c_j u_j \ \rightarrow \max!$$

u.d.N.

$$\begin{array}{lll}
x_{j+1} = x_j - a_j u_j, & & j = 1, 2, \ldots, n \\
x_1 = V & & \\
0 \leq x_{j+1} \leq V, & & j = 1, 2, \ldots, n \\
u_j \in \{0, 1\}, & \text{falls } x_j \geq a_j, & j = 1, 2, \ldots, n \\
u_j = 0, & \text{falls } x_j < a_j, & j = 1, 2, \ldots, n
\end{array}$$

4.2 Problemstellung

Dynamische Optimierungsprobleme betrachten einen endlichen Planungszeitraum, der in n Perioden oder Stufen eingeteilt ist.

Zustandsvariable x_j:

$\rightarrow$ beschreibt Zustand des Systems zu Beginn der Periode j (bzw. am Ende der Periode $j-1$)

$\rightarrow$ $x_1 := x_a$ - vorgegebener Anfangszustand des Systems

Entscheidungsvariable u_j:

$\rightarrow$ In Periode 1 wird Entscheidung u_1 getroffen, die das System in den Zustand x_2 überführt, d.h.

$$x_2 = f_1(x_1, u_1),$$

wobei mit der Entscheidung u_1 die Kosten $g_1(x_1, u_1)$ verbunden sind.

allgemein:

$x_{j+1} = f_j(x_j, u_j)$	resultierender Zustand
$g_j(x_j, u_j)$	Stufenkosten
$X_{j+1} \neq \emptyset$	*Zustandsbereich*, der mögliche Zustände am Ende von Periode j enthält, wobei $X_1 = \{x_1\}$
$U_j(x_j) \neq \emptyset$	*Steuerbereich*, der mögliche Entscheidungen in Periode j enthält (hängt vom Zustand x_j zu Beginn von Periode j ab)

Optimierungsproblem:

$$
\begin{aligned}
&\sum_{j=1}^{n} g_j(x_j, u_j) \;\rightarrow \min! \\
&\text{u.d.N.} \\
&x_{j+1} = f_j(x_j, u_j), \quad j = 1, 2, \ldots, n \\
&x_1 = x_a, \\
&x_{j+1} \in X_{j+1}, \qquad j = 1, 2, \ldots, n \\
&u_j \in U_j(x_j), \qquad j = 1, 2, \ldots, n
\end{aligned}
\tag{4.2}
$$

Bemerkung: Im Allgemeinen wächst der Rechenaufwand bei der Lösung dynamischer Optimierungsprobleme exponentiell mit der Dimension der Steuer- und Zustandsvariablen.

Definition 1:

Eine Folge von Entscheidungen $(u_1, u_2, \ldots, u_n)$ wird *Politik* oder *Steuerung* genannt. Die zu einer gegebenen Politik $(u_1, u_2, \ldots, u_n)$ gemäß

$$x_1 = x_a \text{ und } x_{j+1} = f_j(x_j, u_j) \qquad \text{für alle } j = 1, 2, \ldots, n$$

sukzessiv zu berechnende Folge $(x_1, x_2, \ldots, x_n, x_{n+1})$ heißt zugeordnete *Zustandsfolge*.

Eine Politik oder Zustandsfolge, die den Nebenbedingungen von (4.2) genügt, heißt *zulässig*.

4.3 Bellmansche Funktionalgleichung und Bellmansches Optimalitätsprinzip

Gegeben sind g_j, f_j, X_{j+1} und U_j für alle $j = 1, 2, \ldots, n$.

$\Rightarrow$ Optimierungsproblem hängt von x_1 ab, d.h. $P_1(x_1)$.

analog: $P_j(x_j)$ - Problem für die Perioden $j, j+1, \ldots, n$ mit Anfangszustand x_j

Theorem 1: ***(Bellmansches Optimalitätsprinzip)***

Seien $(u_1^*, \ldots, u_j^*, \ldots, u_n^*)$ eine optimale Politik für das Problem $P_1(x_1)$ und x_j^* der Zustand zu Beginn von Periode j, dann ist

$$(u_j^*, \ldots, u_n^*)$$

eine optimale Politik für das Problem $P_j(x_j^*)$, d.h.:

Die Entscheidungen in den Perioden $j, \ldots, n$ des n-periodigen Problems $P_1(x_1)$ sind (bei gegebenem Zustand x_j^*) unabhängig von den Entscheidungen in den Perioden $1, \ldots, j-1$.

Bellmansche Funktionalgleichungen:

Seien $v_j^*(x_j)$ die minimalen Kosten für das Problem $P_j(x_j)$. Die für $j = 1, 2, \ldots, n$ gültige Beziehung

$$\boxed{\begin{aligned} v_j^*(x_j) &= g_j(x_j, u_j^*) + v_{j+1}^*(x_{j+1}^*) \\ &= \min_{u_j \in U_j(x_j)} \left\{ g_j(x_j, u_j) + v_{j+1}^*[f_j(x_j, u_j)] \right\} \\ x_j &\in X_j \end{aligned}} \tag{4.3}$$

wird *Bellmansche Funktionalgleichung* (BFGL) genannt, wobei

$$v_{n+1}^*(x_{n+1}) = 0$$

für $x_{n+1} \in X_{n+1}$.

$\Rightarrow$ Funktion v_j^* lässt sich bei bekannten v_{j+1}^* berechnen.

BFGL lassen sich auch für die folgenden Fälle angeben:

(a) $\sum_{j=1}^{n} g_j(x_j, u_j) \rightarrow \max!$

$\Rightarrow$ Ersetze in (4.3) min! durch max!

(b) $\prod_{j=1}^{n} g_j(x_j, u_j) \rightarrow \min!$

$\Rightarrow$ BFGL:

$$v_j^*(x_j) = \min_{u_j \in U_j(x_j)} \left\{ g_j(x_j, u_j) \cdot v_{j+1}^*\big[f_j(x_j, u_j)\big] \right\}$$

wobei $v_{n+1}^*(x_{n+1}) := 1$ und

$g_j(x_j, u_j) > 0$ für alle $x_j \in X_j, u_j \in U_j(x_j), j = 1, 2, \ldots, n$

(c) $\max_{1 \leq j \leq n} \big\{ g_j(x_j, u_j) \big\} \rightarrow \min!$

$\Rightarrow$ BFGL:

$$v_j^*(x_j) = \min_{u_j \in U_j(x_j)} \left\{ \max\big\{ g_j(x_j, u_j); v_{j+1}^*\big[f_j(x_j, u_j)\big] \big\} \right\}$$

wobei $v_{n+1}^*(x_{n+1}) = 0$

Bellmansche Funktionalgleichungsmethode

$\Rightarrow$ sukzessive Auswertung der BFGL (4.3) für

$$j = n, n-1, \ldots, 1$$

zur Berechnung von $v_j^*(x_j)$

Algorithmus DO

Schritt 1: Rückwärtsrechnung

(a) Setze $v^*_{n+1}(x_{n+1}) := 0$ für alle $x_{n+1} \in X_{n+1}$.

(b) Für $j = n, n-1, \ldots, 1$ führe aus:

für alle $x_j \in X_j$ bestimme $z^*_j(x_j)$ als Minimalstelle der Funktion

$$w_j(x_j, u_j) := g_j(x_j, u_j) + v^*_{j+1}\big[f_j(x_j, u_j)\big]$$

auf $U_j(x_j)$, d.h.

$$w_j(x_j, z^*_j(x_j)) = \min_{u_j \in U_j(x_j)} w_j(x_j, u_j) = v^*_j(x_j) \text{ für } x_j \in X_j$$

Schritt 2: Vorwärtsrechnung

(a) Setze $x^*_1 := x_a$.

(b) Für $j = 1, 2, \ldots, n$ führe aus:

$$u^*_j := z^*_j(x_j),\ x^*_{j+1} := f_j(x^*_j, u^*_j)$$

$\Rightarrow$ $(u^*_1, u^*_2, \ldots, u^*_n)$ optimale Politik

$\Rightarrow$ $(x^*_1, x^*_2, \ldots, x^*_{n+1})$ optimale Zustandsfolge für Problem $P_1(x^*_1 = x_a)$

Zusammenfassung DO (Dynamische Optimierung)

Phase 1: *Dekomposition*

Phase 2: *Rückwärtsrechnung*

Phase 3: *Vorwärtsrechnung*

Bemerkung: Lassen sich alle Gleichungen

$$x_{j+1} = f_j(x_j, u_j), \quad j = 1, 2, \ldots, n$$

eindeutig nach x_j auflösen, ist der Rechenverlauf umkehrbar, d.h. man kann zunächst eine Vorwärts- und dann eine Rückwärtsrechnung durchführen (z.B. Lagerhaltungsproblem aus 4.1).

4.5 Beispiele und Anwendungen

(a) Das binäre Rucksackproblem

Annahme: V, a_j, c_j - ganzzahlig

$$g_j(x_j, u_j) = c_j u_j, \qquad j = 1, 2, \ldots, n$$

$$f_j(x_j, u_j) = x_j - a_j u_j, \qquad j = 1, 2, \ldots, n$$

$$X_{j+1} = \{0, 1, \ldots, V\}$$

$$U_j(x_j) = \begin{cases} \{0, 1\} & \text{für } x_j \geq a_j \\ 0 & \text{für } x_j < a_j \end{cases}, j = 1, 2, \ldots, n$$

BFGL:

$$v_j^*(x_j) = \max_{u_j \in U_j(x_j)} \{c_j u_j + v_{j+1}^*(x_j - a_j u_j)\}, \quad 1 \leq j \leq n$$

Rückwärtsrechnung:

$$v_n^*(x_n) = \begin{cases} c_n, & \text{falls } x_n \geq a_n \\ 0, & \text{sonst} \end{cases}$$

$$z_n^*(x_n) = \begin{cases} 1, & \text{falls } x_n \geq a_n \\ 0, & \text{sonst} \end{cases}$$

$j = n-1, n-2, \ldots, 1$:

$$v_j^*(x_j) = \begin{cases} \max\{v_{j+1}^*(x_j);\ c_j + v_{j+1}^*(x_j - a_j)\}, & \text{falls } x_j \geq a_j \\ v_{j+1}^*(x_j), & \text{sonst} \end{cases}$$

$$z_j^*(x_j) = \begin{cases} 1, & \text{falls } v_j^*(x_j) > v_{j+1}^*(x_j) \\ 0, & \text{sonst} \end{cases}$$

$\Rightarrow v_1^*(V)$ - maximaler Wert der Rucksackfüllung

Vorwärtsrechnung:

$$\begin{aligned} x_1^* &:= V \\ u_j^* &:= z_j^*(x_j^*), \qquad j = 1, 2, \ldots, n \\ x_{j+1}^* &:= x_j^* - a_j u_j^*, \quad j = 1, 2, \ldots, n \end{aligned}$$

(b) Bestimmung eines kürzesten (längsten) Weges in einem Graphen

Ziel: Bestimme einen kürzesten Weg vom Knoten (Ort) x_1 zum Knoten (Ort) x_{n+1}.

Seien:

$X_j = \{x_j^1, x_j^2, \ldots, x_j^k\}$ - Menge aller Orte der Stufe j, $2 \leq j \leq n$

$X_1 = \{x_1\}, \quad X_{n+1} = \{x_{n+1}\}$

$U_j(x_j) = \{x_{j+1} \in X_{j+1} \mid \exists \text{ Bogen von } x_j \text{ nach } x_{j+1}\}$, $j = 1, 2, \ldots, n$

$v_j^*(x_j)$ - Länge eines kürzesten Weges vom Knoten $x_j \in X_j$ zum Knoten x_{n+1}

$g_j(x_j, u_j) = c_{x_j, u_j}$

$f_{j+1}(x_j, u_j) = u_j = x_{j+1}$

$z_j^*(x_j) = u_j = x_{j+1}$ falls x_{j+1} nächster Ort nach x_j auf einem kürzesten Weg von x_j nach x_{n+1} ist

BFGL:

$$v_n^*(x_n) = c_{x_n,x_{n+1}} \quad \text{für} \quad x_n \in X_n$$

$j = n-1, n-2, \ldots, 1$:

$$v_j^*(x_j) = \min\{c_{x_j,x_{j+1}} + v_{j+1}^*(x_{j+1}) \mid x_{j+1} \in X_{j+1}, \text{ so dass Bogen } (x_j, x_{j+1}) \text{ existiert}\}$$

(c) Personalzuordnung

Beispiel 3: Personalzuordnung

Drei Forschungsteams arbeiten an der Lösung einer Aufgabe. Die Wahrscheinlichkeit eines Misserfolges sei bei den einzelnen Teams mit 0,4, 0,6 und 0,8 gegeben. Zur Erhöhung der Erfolgsaussichten sollen zwei weitere Mitarbeiter eingesetzt werden, wobei die geschätzten Vorteile der erhöhten Mitarbeiterzahl durch die folgende Tabelle der Fehlschlagswahrscheinlichkeiten gegeben sind.

Mitarbeiterzahl	Team 1	Team 2	Team 3
0	0,4	0,6	0,8
1	0,2	0,4	0,5
2	0,15	0,2	0,3

Ziel: Setze die Mitarbeiter so ein, dass die Gesamtwahrscheinlichkeit eines Fehlschlags möglichst klein wird.

Annahme: Die Wahrscheinlichkeiten für einen Misserfolg der einzelnen Teams sind *unabhängig* voneinander.

Seien:

x_j - Anzahl der noch für die Teams $j, \ldots, 3$ zur Verfügung stehenden Mitarbeiter

$x_1 = 2$ und $x_4 = 0$

u_j - Anzahl der dem Team j zugeordneten Mitarbeiter

$W_j(u_j)$ - Fehlschlagswahrscheinlichkeit von Team j bei Einsatz von u_j zusätzlichen Mitarbeitern

Optimierungsproblem:

$$W(u_1) \cdot W(u_2) \cdot W(u_3) \ \rightarrow \min!$$

u.d.N.

$$\begin{aligned}
&u_1 + u_2 + u_3 = 2 \\
&u_j \in \{0, 1, 2\}, \quad j = 1, 2, 3 \\
&x_{j+1} = x_j - u_j, \ j = 1, 2, 3 \\
&x_1 = 2, \quad x_4 = 0
\end{aligned}$$

Bemerkung:

Die Funktionen $g_j(x_j, u_j) = W_j(u_j)$ hängen nur von den Entscheidungen u_j ab.

> $v_j^*(x_j)$ - minimale Fehlschlagswahrscheinlichkeit, falls x_j Mitarbeiter für die Teams $j, \ldots, 3$ zur Verfügung stehen

BFGL:

$$v_j^*(x_j) = \min\{W_j(u_j) \cdot v_{j+1}^*(x_j - u_j) \mid u_j \in \{0, \ldots, x_j\}\}$$

wobei $v_{n+1}^*(0) = 1$

(d) Endlagerung eines Schadstoffes

Für die Endlagerung eines Schadstoffes werden 3 mögliche Deponien $D_1,\ D_2,\ D_3$ in Betracht gezogen. Die gesamte Schadenswirkung je Einheit des Schadstoffes wird für die Deponien durch die Kosten $a_1,\ a_2,\ a_3$ beschrieben.

Ziel: Aufteilung von K Mengeneinheiten des Schadstoffes auf die Deponien, so dass der maximale Schaden möglichst klein wird.

Seien:

x_j - die zur Verteilung auf $D_j, \ldots, D_3$ verbleibende Schadstoffmenge

$x_0 = K$ und $x_4 = 0$

u_j - die auf D_j gelagerte Schadstoffmenge

Optimierungsproblem:

$$\max\{a_1u_1, a_2u_2, a_3u_3\} \rightarrow \min!$$

u.d.N.

$$\begin{aligned} & u_1 + u_2 + u_3 = K \\ & 0 \leq u_j \leq x_j, \qquad j = 1,2,3 \\ & x_{j+1} = x_j - u_j, \quad j = 1,2,3 \\ & x_1 = K, \quad x_4 = 0 \end{aligned}$$

$v_j^*(x_j)$ - minimale Kosten bei Verteilung von x_j Einheiten auf die Deponien $D_j, \ldots, D_3$

BFGL:

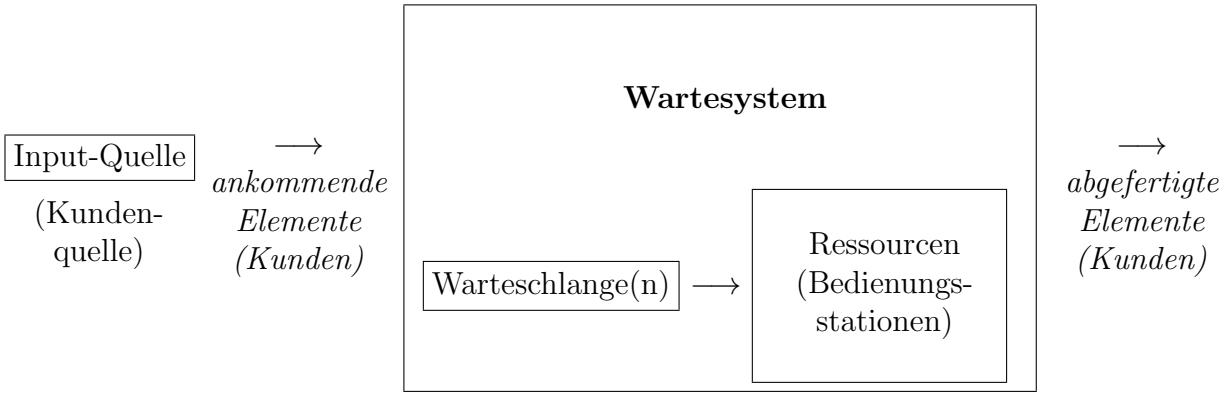
$$v_3^*(x_3) = a_3 x_3 \quad (= a_3 u_3)$$

$$v_j^*(x_j) = \min\left\{ \max(a_j u_j;\ v_{j+1}^*(x_j - u_j)) \ \mid\ 0 \leq u_j \leq x_j \right\}, \qquad j = 1, 2$$

5 Warteschlangen

5.1 Charakterisierung von Wartesystemen

prinzipieller Aufbau eines Wartesystems:



Einige Beispiele:

ankommende Elemente (*Input*)	Warteschlange	Ressource	abgefertigte Elemente (*Output*)
ankommende Telefongespräche	Gespräche in Leitung	Telefonleitung (Zentrale)	hergestellte Verbindungen
ankommende Autos	Autoschlange (Verkehrsstockung)	Kreuzung (Ampel)	abfahrende Autos
ankommende Autos	Autoschlange	Tankstelle	betankte Autos
Maschinendefekte	zu reparierende Maschinen	Reparatur (Monteur)	intakte Maschinen
Fahrgäste	Fahrgast-schlange	Taxis	Fahrgäste auf Fahrt zum Ziel

Die Warteschlangentheorie nutzt stochastische Prozesse:

- Ankunftsprozess
- Bedienungsprozess

abgeleitete stochastische Prozesse:

→ *Warteschlangenprozess:* Zahl der Kunden im System

→ *Wartezeitprozess:* Zeit von Ankunft eines Kunden bis zur Bedienung

→ *Output-Prozess:* Abstand zwischen dem Abschluss zweier aufeinanderfolgender Bedienungen

→ *Betriebsperiode:* Zeit zwischen zwei aufeinanderfolgenden Leerzeiten des Bedienungskanals

Annahmen und Bezeichnungen

Wartesystem zur Zeit 0 leer

$T_1 < T_2 < \ldots$ - Zeitpunkte, zu denen Kunden einzeln an der Bedienungsstation ankommen

$Z_n = T_n - T_{n-1}, \quad n = 1, 2, \ldots, \quad T_0 := 0$ - Zwischenankunftszeit des n-ten Kunden

S_n - Bedienungs- oder Servicezeit des n-ten Kunden

W_n^q - Wartezeit des n-ten Kunden in der Schlange

$W_n := W_n^q + S_n$ - Verweilzeit des n-ten Kunden

$D_n = T_n + W_n^q + S_n$ - n-ter Kunde verlässt Wartesystem

$\mathcal{V}(t)$ - virtuelle Wartezeit: Wartezeit eines Kunden in der Schlange, der zum Zeitpunkt t ankommen würde, d.h. $\mathcal{V}(T_n) = W_n^q$

$\mathcal{L}(t)$ - Anzahl der Kunden im Wartesystem zur Zeit t

$\mathcal{L}^q(t)$ - Anzahl der Kunden in der Schlange zur Zeit t

$\Rightarrow T_n,\ Z_n,\ S_n,\ W_n^q,\ W_n,\ \mathcal{V}(t),\ \mathcal{L}(t),\ \mathcal{L}^q(t)$ sind *Zufallsgrößen*!

Annahmen über Zufallsgrößen:

Z_n unabhängig, identisch verteilt

$\Rightarrow$ Zufallsgröße Z (*Zwischenankunftszeit*)

S_n unabhängig, identisch verteilt $\Rightarrow$ Zufallsgröße S (*Bedienungszeit*)

Z und S unabhängig voneinander

Charakteristika von Wartesystemen:

- Größe des Warteraums:
  - endlich
  - unendlich
- Anzahl der Bedienungsschalter
- Abfertigungsmodus:
  - einzeln
  - schubweise
- Warteschlangendisziplin:
  - *first come first served (FCFS)*
  - last come first served
  - zufällige Auswahl

Beschreibung von Wartesystemen

3-Tupel $x \mid y \mid z$

x: Verteilung der Zwischenankunftszeit

y: Verteilung der Bedienungszeit

z: Anzahl der parallelen Bedienungsschalter

Ausprägungen von x und y:

M (Markov)	Exponentialverteilung mit Dichtefunktion $f(t) = \alpha e^{-\alpha t}, \quad t \geq 0, \ \alpha > 0$
E_k	Erlangverteilung mit Phasenparameter k und Dichtefunktion $f_k(t) = \frac{b^{k+1} t^k e^{-bt}}{k!}, \quad t, b \geq 0, \ k = 0, 1, 2, \ldots$
G (general)	beliebige Verteilung
D	Verteilung deterministischer Größe, z.B. $P(t = a) = 1$

Beispiel: $M \mid M \mid 1$

$\rightarrow Z, S$ exponentialverteilte Zufallsgrößen, 1 Bedienungsschalter

Bei Beschränkungen der Kanäle oder der Zahl der zu berücksichtigenden Input-Elemente nutze:

$$x \mid y \mid z \mid a \mid b,$$

wobei:

a - *begrenzte Kapazität des Systems* (Schlangenkapazität und 1 Element pro Kanal)

b - *Zahl der (endlich vielen) Input-Elemente*

Beispiel: $M \mid M \mid s \mid\mid K$

$\rightarrow Z, S$ exponentialverteilte Zufallsgrößen

s Bedienungsschalter

K relevante Input-Elemente

5.2 Das System $M \mid M \mid 1$

Annahmen:

1. Z ist exponentialverteilt.

   $P(Z \leq t) = 1 - e^{-\alpha t}$ Verteilungsfunktion

   $f_Z(t) = \alpha e^{-\alpha t}$ Dichtefunktion

   $E(Z) = \frac{1}{\alpha}$ Erwartungswert von Z

   α - Ankunftsrate

   $\Rightarrow$ Ankünfte können durch einen Poisson-Prozess mit dem Parameter α beschrieben werden.

2. S ist exponentialverteilt.

   $P(S \leq t) = 1 - e^{-\beta t}$ Verteilungsfunktion

   $f_S(t) = \beta e^{-\beta t}$ Dichtefunktion

   $E(S) = \frac{1}{\beta}$ Erwartungswert von S

   β - Bedienungsrate

3. unbegrenzter Warteraum, Kunden werden gemäß FCFS bedient

Ereignis: Eintreffen oder Abfertigen eines Kunden

$$\varrho := \frac{\alpha}{\beta} \quad \text{Verkehrsintensität}$$

Zur Anzahl der Kunden im System

Zustandswahrscheinlichkeit:

$$P_n(t) = P(\mathcal{L}(t) = n)$$

Konvergenzbedingung

$\alpha < \beta$ $(\varrho < 1)$

$\Rightarrow$ Zustandwahrscheinlichkeit $P_n(t)$ konvergieren für $t \to \infty$ gegen stationäre Werte P_n

Rekursive Lösung der stationären Punkte liefert:

$$P_n = \left(\frac{\alpha}{\beta}\right)^n \cdot P_0 = \varrho^n \cdot P_0$$

außerdem gilt:

$$\sum_{n=0}^{\infty} P_n = \sum_{n=0}^{\infty} \varrho^n P_0 = \frac{1}{1-\varrho} P_0 = 1$$
$$\Rightarrow P_0 = 1 - \varrho, \qquad P_n = \varrho^n (1-\varrho) :$$

geometrisch verteilt mit Parameter $\varrho = \frac{\alpha}{\beta}$

Beispiel 1: $M \mid M \mid 1$

An der Kasse eines Supermarktes werden durchschnittlich 25 Kunden pro Stunde bedient, und die durchschnittliche Bedienungsdauer pro Kunde beträgt 2 Minuten.

Annahmen: Z, S exponentialverteilt, System im stationären Zustand (*Gleichgewichtsfall*)

(a) Wie groß ist der Anteil der Leerzeiten der Kassiererin?

(b) Wie groß ist die Wahrscheinlichkeit, dass mehr als 5 Kunden an der Kasse stehen?

Mittlere Anzahl der Kunden im System L/ Mittlere Schlangenlänge L^q

$$L := E(\mathcal{L}) = \sum_{n=0}^{\infty} nP_n = \sum_{n=0}^{\infty} n(1-\varrho)\varrho^n = \frac{\varrho}{1-\varrho}$$

$$L = \frac{\alpha}{\beta - \alpha} \tag{6}$$

$$L^q = E(\mathcal{L}^q) = \sum_{n=1}^{\infty}(n-1)P_n = L - (1-P_0) = \frac{\varrho^2}{1-\varrho}$$

$$L^q = \frac{\alpha^2}{\beta(\beta-\alpha)} = L - \frac{\alpha}{\beta}$$

Mittlere Warte- und Verweilzeit der Kunden

Sei $F(\cdot, t)$ die Verteilungsfunktion der virtuellen Wartezeit $\mathcal{V}(t)$

$$\Rightarrow \quad F(v,t) = P(\mathcal{V}(t) \leq v) = \sum_{n=0}^{\infty} P(\mathcal{V}(t) \leq v \mid \mathcal{L}(t) = n) \cdot P(\mathcal{L}(t) = n)$$

$$\Rightarrow \quad \lim_{t \to \infty} F(v,t) := F_q(v) = \begin{cases} 1 - (\frac{\alpha}{\beta}) \cdot e^{-(\beta-\alpha)v} & , \text{falls } v \geq 0 \\ 0 \ , & \text{falls } v < 0 \end{cases}$$

$$W^q := E(\mathcal{W}^q) = \frac{\alpha}{\beta(\beta - \alpha)}$$

$\rightarrow W^q$: mittlere Wartezeit eines Kunden in der Schlange im Gleichgewichtsfall

analog:

$$\text{Sei } F(w) = P(\mathcal{W} < w) = \begin{cases} 1 - e^{-(\beta-\alpha)w} & , \text{falls } w \geq 0 \\ 0 \ , & \text{falls } w < 0 \end{cases}$$

$$W = E(\mathcal{W}) = W^q + \frac{1}{\beta} = \frac{1}{\beta - \alpha} \qquad (7)$$

$\rightarrow W$: mittlere Verweilzeit eines Kunden im Gleichgewichtsfall

Aus Beziehung (6) und (7) folgt:

$$L = \alpha \cdot W$$

analog: $L^q = \alpha \cdot W^q$

Little's Formel

Beispiel 2: $M \mid M \mid 1$

Betrachtet wird Beispiel 1.

(a) Wie lange muss ein Kunde durchschnittlich an der Kasse warten?

(b) Wie viele Kunden müssen länger als 10 Minuten verweilen?

5.3 Das System $M \mid M \mid 1 \mid K$ mit endlichem Warteraum

- maximal K Kunden im System ($K-1$ Kunden in der Warteschlange)

- Gleichgewichtsfall

$$P_n = \begin{cases} \dfrac{1}{1+K} & \text{für } \varrho = 1 \\ \dfrac{(1-\varrho)\varrho^n}{1-\varrho^{K+1}} & \text{für } \varrho \neq 1 \end{cases}$$

$$n = 0, 1, \ldots, K$$

- Sei γ der *Erfassungsgrad der Kunden* (Anteil der Kunden, die sich in die Warteschlange einreihen)

$$\boxed{\Rightarrow \quad \gamma = 1 - P_K}$$

mittlere Anzahl der Kunden im System:

$$L := E(\mathcal{L}) = \frac{\varrho[1-(K+1)\varrho^K + K\varrho^{K+1}]}{(1-\varrho^{K+1})(1-\varrho)} \quad (\varrho \neq 1)$$

mittlere Anzahl der Kunden in der Warteschlange:

$$L^q = L - \frac{\alpha\gamma}{\beta}$$

mittlere Verweilzeit im System:

$$W = \frac{L}{\alpha\gamma} = \frac{L}{\alpha(1-P_K)} \qquad \text{(Little's Formel)}$$

mittlere Wartezeit in der Schlange:

$$W^q = W - \frac{1}{\beta}$$

Beispiel 3: $M \mid M \mid 1 \mid K$

Bei Dr. Dent treffen durchschnittlich 5 Patienten pro Stunde ein, und er kann durchschnittlich 6 Patienten pro Stunde behandeln.

(a) Es liege ein $M \mid M \mid 1$ System vor. Wie groß ist die erwartete Anzahl von Patienten in der Praxis? Wie lange muss ein Patient durchschnittlich warten? Welcher Anteil von Patienten muss nicht warten?

(b) Im Wartesystem befinden sich 4 Stühle, wobei ein Stuhl im Behandlungszimmer und 3 Stühle im Wartezimmer stehen. Es wird angenommen, dass Patienten nicht warten, wenn alle Stühle besetzt sind. Wie groß sind die erwartete Anzahl Patienten in der Praxis und die durchschnittliche Wartezeit für dieses System? Wie viele Patienten verliert Dr. Dent pro Stunde durch die Begrenzung des Warteraumes?

5.4 Das System $M \mid M \mid s$

- $s > 1$ parallele, identische Bedienungsschalter
- Gleichgewichtsfall
- Konvergenzbedingung: $\varrho = \frac{\alpha}{\beta} < s$

$$P_n = \begin{cases} \dfrac{\varrho^n}{n!} \cdot P_0 & \text{für } n = 1, 2, \ldots, s-1 \\ \dfrac{\varrho^n}{s! \cdot s^{n-s}} \cdot P_0 & \text{für } n \geq s \end{cases}$$

mit

$$P_0 = \frac{1}{1 + \sum_{j=1}^{s-1} \frac{\varrho^j}{j!} + \frac{\varrho^s}{(s-\varrho)(s-1)!}}$$

mittlere Anzahl der Kunden in der Warteschlange:

$$L^q = \frac{\varrho^{s+1}}{(s-\varrho)^2(s-1)!} \cdot P_0$$

mittlere Anzahl der Kunden im System:

$$L = L^q + \varrho$$

mittlere Wartezeit in der Schlange:

$$W^q = \frac{L^q}{\alpha}$$

mittlere Verweilzeit im System

$$W = \frac{L}{\alpha} = \frac{L^q + \frac{\alpha}{\beta}}{\alpha} = W^q + \frac{1}{\beta}$$

Beispiel 4: $M \mid M \mid s$

In einer Telefonzelle werden im Mittel 10 Gespräche pro Stunde geführt, deren erwartete Dauer jeweils 5 Minuten sei. Die zumutbare mittlere Wartezeit vor der Zelle sei 5 Minuten. Wie viele Telefonzellen sind erforderlich? (Annahmen: Z, S exponentialverteilt; Gleichgewichtsfall)

5.5 Das System $M \mid M \mid s \mid K$ mit endlichem Warteraum

- im Wartesystem haben höchstens $K \geq s$ Kunden Platz
- Gleichgewichtsfall

$$P_n = \begin{cases} \dfrac{\varrho^n}{n!} \cdot P_0 & \text{für } n = 1, 2, \ldots, s-1 \\ \dfrac{\varrho^n}{s! \cdot s^{n-s}} \cdot P_0 & \text{für } n = s, s+1, \ldots, K \\ 0 & \text{für } n > K \end{cases}$$

mit

$$P_0 = \frac{1}{1 + \sum\limits_{j=1}^{s-1} \frac{\varrho^j}{j!} + \frac{\varrho^s}{s!} \cdot \sum\limits_{j=s}^{K} \left(\frac{\varrho}{s}\right)^{j-s}}$$

mittlere Anzahl der Kunden in der Warteschlange:

$$L^q = \frac{\varrho^{s+1} P_0}{(s-\varrho)^2 (s-1)!} \cdot \left[1 - (\frac{\varrho}{s})^{K-s} \cdot \big(1 + \frac{(s-\varrho)(K-s)}{s}\big) \right] \quad (\varrho \neq s)$$

mittlere Wartezeit in der Schlange:

$$W^q = \frac{L^q}{\alpha \cdot \gamma}, \qquad \alpha \cdot \gamma = \alpha_{eff} \qquad \text{mit Erfassungsgrad } \gamma = 1 - P_K$$

mittlere Verweilzeit im System:

$$W = W^q + \frac{1}{\beta}$$

mittlere Anzahl der Kunden im System:

$$L = \alpha_{eff} W = L^q + \frac{\alpha_{eff}}{\beta}$$

Beispiel 5: $M \mid M \mid s \mid K$

Ein Reisebüro habe 2 Angestellte, die Anrufe beantworten, und ein Anrufer kann zusätzlich in der 'Warteleitung' bleiben. Sind alle 3 Leitungen belegt, hört der Anrufer das Besetztzeichen. Es seien $\alpha = 1$ und $\beta = 2$ (pro Minute) sowie Z, S exponentialverteilt. Gesucht sind die stationären Wahrscheinlichkeiten, dass

(a) ein Anrufer sofort mit einem Angestellten sprechen kann;

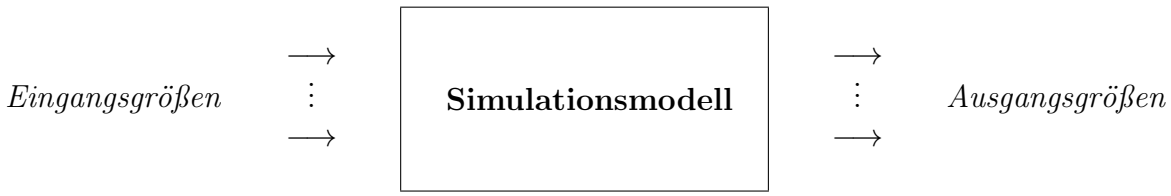
(b) ein Anrufer zunächst in der Warteleitung bleibt;

(c) ein Anrufer das Besetztzeichen hört?

6 Simulation

6.1 Grundbegriffe und Beispiele

Simulation: Nachbilden von Prozessen realer Systeme in einem Modell und anschließendes Durchführen von Experimenten am Modell



Simulation eines Wartesystems:

variable Eingangsgrößen:	Ankunftszeit der Kunden
Parameter:	mittlere Anzahl ankommender Kunden pro Zeiteinheit
	mittlere Bedienungszeit
Ausgangsgrößen:	mittlere Warteschlangenlänge
	maximale Bedienungszeit eines Kunden

$\rightarrow$ oft Nutzung der *Digitalsimulation* (stetige Abläufe oder Größen werden durch diskrete Abläufe/Größen approximiert)

Arten der Simulation

deterministische Simulation	Eingangsgrößen legen Ausgangsgrößen eindeutig fest (deterministisches Modell)
stochastische Simulation	Zufallseinflüsse werden über Zufallsvariable erfasst (stochastisches Modell)

Stufen einer Simulationsstudie

- *Problemformulierung*
- *Entwicklung eines Simulationsmodells*
- *Datenerhebung und Datengenerierung*

  (1) Welche Verteilungsgesetze gelten?

  (2) Wie werden Realisationen von Zufallsgrößen erzeugt?

  zu (1): Nutze Methoden zur Ermittlung von Konfidenzintervallen von Verteilungsparametern und Tests zum Überprüfen von Hypothesen über die Lage von Verteilungsparametern.

  zu (2): z.B. lineare Kongruenzgeneratoren, inverse Transformationsmethode

- *Erstellung eines Computerprogramms*
  (auch Nutzung kommerzieller Simulations-Software, siehe Abschnitt 6.2)
- *Modellvalidierung*
  (Überprüfung der Gültigkeit des Modells mit untersuchtem Realitätsausschnitt)
- *Planung und Durchführung von Simulationsläufen*

  (1) Wie ist der Anfangszustand für die Simulationsläufe zu wählen?

(2) Wie lässt sich der Simulationsumfang bestimmen?

zu (1): Verwendung eines 'typischen' Anfangszustandes oder des 'Leerzustandes'

zu (2): z.B. bezogen auf den Mittelwert einer abhängigen Zufallsgröße:

- Schätzung des Mittelwertes
- Konfidenzintervall für den Mittelwert
- Ermittlung des Stichprobenumfangs bei gegebener Genauigkeit
  (vorgegebene Irrtumswahrscheinlichkeit)

- *Auswertung der Simulationsstudie*

6.2 Einige Bemerkungen zur Nutzung von Simulationssoftware

- **Nutzung von 'Spreadsheets'**
- **EXCEL Queueing Simulator** für Warteschlangen

  Eingabe:

  - Anzahl der Bedienungsstationen
  - Zwischenankunftszeit T

    z.B. Exponentialverteilung (Erwartungswert), Erlangverteilung (Erwartungswert, k), $[a, b]$-Gleichverteilung (Parameter a, b)
  - Bedienungszeit S analog zu T
  - Simulationsumfang

  Ausgabe:

  $L,\ L^q,\ W,\ W^q,\ P_0,\ P_1, \ldots,\ P_{10}$

  als Punktschätzung und als 95% Konfidenzintervall

  $\rightarrow$ *sehr einfache Nutzung*
- **Crystal Ball** 2000.5 Studentenversion (140 Tage)

  4 Schritte:

  - Definition der zufälligen Eingangsgrößen (z.B. 17 verschiedene Verteilungsfunktionen)

  - Definition der Ausgangsgrößen
  - Setzen von Präferenzen (z.B. Simulationsumfang)
  - Ausführen der Simulation

  weitere Erläuterungen: siehe Hillier/Lieberman, Abschnitt 20.6 bzw.

  http://www.oracle.com/appserver/business-intelligence/crystalball/index.html

- **RiskSim**
  - akademische Version: siehe CD-ROM in Hillier/Lieberman (Shareware)
  - nicht so vielseitig wie Crystal Ball, aber einfach zu nutzen