Prof. Dr. Frank Werner

Fakultät für Mathematik

Institut für Mathematische Optimierung

http://math.uni-magdeburg.de/∼werner/or-ma.html

Operations Research

Vorlesungsskript
(auszugsweise)

Sommersemester 2019

Bemerkungen:

1. Dieses Skript ersetzt nicht den Besuch der Vorlesung. Zahlenbeispiele werden ausschließlich in der Vorlesung präsentiert.
2. Das Symbol ✏ weist auf zusätzliche, detaillierte Ausführungen in der Vorlesung hin.
3. Für die Unterstützung bei der Aufbereitung dieses Vorlesungsskriptes bedanke ich mich bei Frau Julia Lange.

Inhaltsverzeichnis

1 Diskrete Optimierung **1**
- 1.1 Grundbegriffe und Beispiele . . . 1
- 1.2 Lösungsmethoden . . . 5
  - 1.2.1 Definitionen und Verfahrensklassen . . . 5
  - 1.2.2 Branch and Bound Verfahren (B&B) . . . 5
  - 1.2.3 Schnittebenenverfahren . . . 9
  - 1.2.4 Heuristische Verfahren . . . 10
  - 1.2.5 Software-Pakete zur Lösung ganzzahliger und binärer, linearer Optimierungsprobleme . . . 11
- 1.3 Das Rucksackproblem . . . 12

2 Metaheuristiken **15**
- 2.1 Iterative Verfahren, Grundbegriffe . . . 15
- 2.2 Simulated Annealing . . . 17
- 2.3 Tabu-Suche . . . 18
- 2.4 Genetische Algorithmen . . . 19

3 Dynamische Optimierung **21**
- 3.1 Einführungsbeispiele . . . 21
  - 3.1.1 Das Lagerhaltungsproblem . . . 21
  - 3.1.2 Das binäre Rucksackproblem . . . 22
- 3.2 Problemstellung . . . 23
- 3.3 Bellmansche Funktionalgleichung und Bellmansches Optimalitätsprinzip . . . 24
- 3.4 Bellmansche Funktionalgleichungsmethode . . . 25
- 3.5 Beispiele und Anwendungen . . . 27
  - 3.5.1 Das binäre Rucksackproblem . . . 27
  - 3.5.2 Bestimmung eines kürzesten (längsten) Weges in einem Graphen . . . 28
  - 3.5.3 Personalzuordnung . . . 28
  - 3.5.4 Endlagerung eines Schadstoffes . . . 29

4 Warteschlangen **31**
- 4.1 Charakterisierung von Wartesystemen . . . 31
- 4.2 Das System M|M|1 . . . 33
- 4.3 Das System M|M|1|K mit endlichem Warteraum . . . 37
- 4.4 Das System M|M|s . . . 38
- 4.5 Das System M|M|s|K mit endlichem Warteraum . . . 39

5 Simulation **40**
- 5.1 Grundbegriffe und Beispiele . . . 40

5.2 Erzeugung von Zufallszahlen 43
5.3 Einige Bemerkungen zur Nutzung von Simulationssoftware 44

Kapitel 1

Diskrete Optimierung

1.1 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.

Beispiele:

Beispiel 1: Zuordnungsproblem ✏

BEISPIEL 2: INVESTITIONSPLANUNG ✏

Ein Unternehmen zieht 5 Projekte mit den folgenden Aufwendungen (in Mio. €) 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:

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

u.d.N.

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

Illustration der Bereiche ✏

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

u.d.N.

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

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

$\Rightarrow$ 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.

Beispiele (fortgesetzt):

Beispiel 4: Runden oder Abschneiden der optimalen Lösung eines LOP ✏

1.2 Lösungsmethoden

1.2.1 Definitionen und Verfahrensklassen

exakte Verfahren: Bestimmung einer optimalen Lösung in endlich vielen Schritten

vollständige Enumeration: nur bei *sehr kleinen* Problemen möglich

Verfahrensklassen:

- *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.
- *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.
- *Dynamische Optimierung* (siehe Kapitel 3)
- *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 2)
  - Verkürzte exakte Verfahren: z.B. vorzeitig abgebrochene Branch and Bound Verfahren

1.2.2 Branch and Bound Verfahren (B&B)

Betrachtet wird zunächst ein *binäres Optimierungsproblem.*

$$f(\mathbf{x}) \rightarrow \min!$$

u.d.N.

$$\mathbf{x} \in S \subseteq \{0,1\}^n$$

Bemerkung: Bei vollständiger Enumeration ergäben sich beispielsweise für $n = 50$ bereits

$$| \{0,1\}^{50} | = 2^{50} \approx 10^{15}$$

mögliche Kombinationen.

Zustände von Variablen

Variable u_j beschreibt den Zustand von x_j wie folgt:

Zustand von x_j	Wert von x_j	Wert von u_j
fixiert "gesetzt"	1	1
fixiert "gesperrt"	0	0
frei	$0 \vee 1$	-1

- Vektor $\mathbf{u} \in U := \{-1, 0, 1\}^n$ wird mit Knoten u im Verzweigungsbaum identifiziert. Knoten u schränkt die Lösungsmenge wie folgt ein:

$$S(u) = \{\mathbf{x} \in S \mid x_j = u_j,\ x_j \text{ fixiert}\}, \quad j \in \{1, \ldots, n\}$$

- Dem Knoten u entspricht folgendes Optimierungsproblem.

$$\left.\begin{array}{rl} & f(\mathbf{x}) \to \min! \\ \text{u.d.N.} & \\ & \mathbf{x} \in S(u) \end{array}\right\} \quad P(u)$$

Sei $f^*(u) := \min f(\mathbf{x})$ mit $\mathbf{x} \in S(u)$.

Einführung von Schrankenfunktionen

Definition 1:
Eine Funktion $LB: U \to \mathbb{R} \cup \{\infty\}$ heißt *untere Schrankenfunktion*, wenn gilt

(a) $LB(u) \leq \min f(\mathbf{x}) = f^*(u)$ mit $\mathbf{x} \in S(u)$

(b) $S(u) = \{\mathbf{x}\} \Rightarrow LB(u) = f(\mathbf{x})$

(c) $S(u) \subseteq S(v) \Rightarrow LB(u) \geq LB(v)$

Definition 2:
$UB \in \mathbb{R}$ heißt *obere Schranke* für den optimalen Zielfunktionswert, falls $UB \geq \min f(\mathbf{x})$ mit $\mathbf{x} \in S$ gilt.

$\bar{\mathbf{x}}$ repräsentiere die beste bisher gefundene Lösung.

Zu Beginn eines B&B Verfahrens wird $UB := f(\bar{\mathbf{x}})$ gesetzt, wenn $\bar{\mathbf{x}}$ als heuristische Lösung bekannt ist, oder man setzt $UB := \infty$.

Generierung und Abarbeitung des Verzweigungsbaumes

aktiver Knoten: ein Knoten, der noch *nicht* untersucht wurde

Zu Beginn enthält der Verzweigungsbaum nur die Wurzel $u = (-1, -1, \ldots, -1)^T$ als aktiven Knoten.

Untersuchung eines aktiven Knotens u

- *Fall 1:* $LB(u) \geq UB$

  Knoten u wird aus Verzweigungsbaum entfernt, da gemäß Def. 1 (a)

  $$\min f(\mathbf{x}) \geq LB(u) \geq UB \text{ mit } \mathbf{x} \in S(u)$$

  gilt.
  $\rightarrow$ Problem ist ausgelotet.

- *Fall 2a:* $LB(u) < UB$ mit $u_j \in \{0, 1\}$ für $j = 1, 2, \ldots, n$

  Lösung $x = u$ eindeutig bestimmt
  Gilt $\mathbf{x} \in S \quad \Rightarrow$ Wegen

  $$f(\mathbf{x}) = LB(u) < UB = f(\bar{\mathbf{x}}),$$

  ist eine neue beste Lösung gefunden. Setze $\bar{\mathbf{x}} := \mathbf{x}$ und $UB := f(\bar{\mathbf{x}})$. (Der Knoten u ist nicht mehr aktiv.)
  $\rightarrow$ Problem ist ausgelotet.

- *Fall 2b:* $LB(u) < UB$ mit $u_j = -1$ für (mindestens) ein $j \in \{1, 2, \ldots, n\}$

  Erzeugung der Nachfolgerknoten w^i des Knotens u durch Fixierung einer (oder mehrerer) freier Variablen. (Knoten u ist nicht mehr aktiv, aber die Nachfolger w^i von u sind aktiv.)
  $\rightarrow$ Problem wird verzweigt.

Erzeugung eines Verzweigungsbaumes durch Fixierung jeweils einer freien Variablen in der Reihenfolge $x_1, x_2, \ldots, x_n$ liefert den folgenden Binärbaum. ✏

Suchstrategien - Auswahl des nächsten aktiven Knoten zur Untersuchung

(a) *FIFO-Strategie* (first in, first out)

Neu erzeugte Knoten werden ans Ende der Schlange angefügt und der Knoten an erster Stelle der Schlange wird als nächster untersucht.
$\rightarrow$ Breitensuche

(b) *LIFO-Strategie* (last in, first out)

Neu erzeugte Knoten werden ans Ende der Schlange angefügt und der an letzter Stelle befindliche Knoten wird als nächster untersucht.
$\rightarrow$ Tiefensuche

(c) *LLB-Strategie* (least lower bound)

Der Knoten mit der kleinsten $LB(u)$ wird als nächster untersucht.
(Falls $LB(u) \geq UB$ folgt Abbruch.)

Die LIFO-Strategie liefert meist am schnellsten zulässige Lösungen. Im Verlauf der Suche ist oft der Übergang zur FIFO- oder LLB-Strategie empfehlenswert. **Zur Schrankenfunktion** $LB(u)$

Einige oder mehrere Restriktionen von $P(u)$ werden „gelockert“ oder entfernt.
$\Rightarrow$ Daraus erhält man ein einfacheres Problem $P^*(u)$ mit $S^*(u) \supseteq S(u)$.

$P^*(u) \rightarrow$ Relaxation von $P(u)$

Setze $LB(u) := f(\mathbf{x}^*(u))$, wobei $\mathbf{x}^*(u)$ optimale Lösung von $P^*(u)$ ist.

Binäres Problem: Ersetze für die freien Variablen $x_i \in \{0,1\}$ durch $0 \leq x_i \leq 1$.
(*LP-Relaxation*)

B&B-Verfahren für binäre Optimierungsprobleme (Minimierung)

Schritt 1:

- Falls eine zulässige Lösung $\mathbf{x} \in S$ bekannt ist, setze

$$\overline{\mathbf{x}} := \mathbf{x} \quad \text{und} \quad UB := f(\mathbf{x}),$$

  andernfalls setze

$$UB := \infty.$$

- Setze $u^0 := (-1, -1, \ldots, -1)^T$ und $U_a := \{u^0\}$. (u^0 ist Wurzel)

Schritt 2:

- Ist $U_a = \emptyset$, so gehe zu Schritt 4.
  Andernfalls wähle unter Nutzung einer Suchstrategie ein $u \in U_a$, entferne u aus U_a und berechne $LB(u)$.

Schritt 3:

- Gilt $LB(u) \geq UB$, so gehe im Fall der LLB-Strategie zu Schritt 4. Andernfalls eliminiere u aus dem Verzweigungsbaum.
- Ist $LB(u) < UB$ und sind alle Variablen fixiert, so setze im Fall $\mathbf{x} \in S$:

$$\overline{\mathbf{x}} := \mathbf{x} \quad \text{und} \quad UB := f(\mathbf{x}).$$

- Ist $LB(u) < UB$ und mindestens eine Variable frei, dann erzeuge durch Fixierung einer (oder mehrerer) freier Variablen die Nachfolger von u. Füge die Nachfolger von u zu U_a und zum Verzweigungsbaum hinzu.
- Gehe zu Schritt 2.

Schritt 4: (Abbruch)

- Gilt $UB < \infty$, so ist $\overline{\mathbf{x}}$ eine optimale Lösung mit $f(\mathbf{x}) = UB$. Andernfalls hat das Problem keine zulässige Lösung.

Das Vorgehen lässt sich auf gemischt-binäre Probleme der Form

$$f(\mathbf{x}, \mathbf{y}) \to \min!$$

u.d.N.

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \in S$$
$$\mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \{0,1\}^k$$

übertragen. Sind alle binären Variablen fixiert, liegt ein LOP in den Variablen $x_1, x_2, \ldots, x_n$ vor.

Beispiel 5: Binäres Maximierungsproblem ✏

Modifikationen für ganzzahlige Optimierungsprobleme

Verwende als Relaxation das entstehende LOP, wenn $x_i \in \mathbb{Z}_+$ durch $x_i \geq 0$ ersetzt wird (*LP-Relaxation*). Die optimale Lösung (OL) liefert eine untere Schranke $LB(u)$ für Knoten u (es gilt $S^*(u) \supseteq S(u)$).

Verfahren von Dakin: (Verzweigungsstrategie)

Ist in der OL des LOP (mindestens) eine Variable x_i^* nicht ganzzahlig, erzeuge *zwei* Nachfolgerknoten v^k und v^l durch Hinzufügen der folgenden Bedingungen:

$$x_i \leq [x_i^*] \text{ in } S(v^k) \text{ bzw.}$$
$$x_i \geq [x_i^*] + 1 \text{ in } S(v^l)$$

Beispiel 6: Verfahren von Dakin ✏

1.2.3 Schnittebenenverfahren

Verfahren von Gomory:

Füge Schnittbedingungen hinzu, die die OL des jeweils betrachteten LOP 'abschneiden' (**Gomory-Schnitt**). Hierbei darf keine ganzzahlige Lösung verloren gehen.

Illustration ✏

Angenommen, die Basisdarstellung der OL der LP-Relaxation hat die folgende Gestalt:

$$\begin{array}{rcl}
r_{11}x_1 + r_{12}x_2 + \cdots + r_{1n}x_n + x_{n+1} & & = k_1 \\
r_{21}x_1 + r_{22}x_2 + \cdots + r_{2n}x_n + x_{n+2} & & = k_2 \\
\vdots & \ddots & = \vdots \\
r_{m1}x_1 + r_{m2}x_2 + \cdots + r_{mn}x_n + x_{n+m} & & = k_m \\
g_1x_1 + g_2x_2 + \cdots + g_nx_n + Z & & = c
\end{array}$$

Optimale Lösung: $\mathbf{x} = (0, \ldots, 0, k_1, k_2, \ldots, k_m)^T$ mit $g_i \geq 0$ für $i = 1, \ldots, n$

- $x_{n+i} = k_i$ sei eine Variable, die die Ganzzahligkeitsforderung nicht erfüllt.
- Sei $[k] \in \mathbb{Z}$ mit $k - 1 < [k] \leq k$, bestimme:
  $\overline{k_i} := k_i - [k_i]$, $\overline{r_{ij}} = r_{ij} - [r_{ij}]$ für $j = 1, 2, \ldots, n$.
- Führe eine zusätzliche Nebenbedingung in die Basisdarstellung ein.
  → *Gomory-Schnitt*

  $$-\overline{r_{i1}}x_1 - \overline{r_{i2}}x_2 - \ldots - \overline{r_{in}}x_n + \overline{x_{n+i}} = -\overline{k_i}$$

  $\overline{x_{n+i}} \geq 0$ ist neue Schlupfvariable.
- Die ursprüngliche OL ist jetzt wegen $-\overline{k_i} < 0$ unzulässig!

⇒ Bestimme mittels *dualer Simplexmethode* eine neue zulässige Basislösung. Existieren weitere Variable, die die Ganzzahligkeitsbedingung verletzen, führe einen weiteren Gomory-Schnitt durch.

Beispiel 7: Gomory-Schnitt ✏

Betrachtet wird des Zuschnittproblem aus Beispiel 3 (Seite 3).

1.2.4 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)

Algorithmus von Kruskal

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

Algorithmus zur Bestimmung eines Minimalgerüstes

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.

Beispiel 8: Algorithmus von Kruskal ✏

(b) Iterative Verfahren (Verbesserungsverfahren)

→ siehe Kapitel 2

1.2.5 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: pmenge(i) ≤ 30 für alle $i = 1, \ldots, 10$
DATA:
pcost = 100,50,80,120,30,50,10,60,20,90;
ENDDATA
→ Vektor pcost = $(100, 50, 80, 120, 30, 50, 10, 60, 20, 90)^T$ gesetzt

Generierung des Problems
LINGO → Generate → Display Model

Lösen des Problems
LINGO → Solve

Beispielmodellierung

Betrachtet wird das folgende ganzzahlige Problem der Produktionsplanung:

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

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

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

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

1.3 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.

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

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

Beispiel 9: Anwendung des Greedy-Algorithmus beim Rucksackproblem ✏

Einige Bemerkungen zum B&B Algorithmus beim Rucksackproblem

1. Vergleiche Abschnitt 1.2.2 (B&B Algorithmus für binäre Optimierungsprobleme, aber: Maximierungsproblem).
2. Ermittle mittels Greedy-Algorithmus eine zulässige Lösung $\mathbf{x}^G$ und setze $\bar{\mathbf{x}} := \mathbf{x}^G$ und $LB = f_G$.
3. Zur Berechnung oberer Schranken $UB(u)$ wird die LP-Relaxation verwendet, d.h. ersetze die freien Variablen $x_j \in \{0,1\}$ durch $0 \leq x_j \leq 1$.
   Beispielrechnung ✏
4. Gegebenenfalls kann die Dimension des Ausgangsproblems reduziert werden.

   Theorem 1: Seien f_{LP} der optimale Zielfunktionswert der LP Relaxation und f_G der Zielfunktionswert des Greedy-Algorithmus für das Rucksackproblem (R) sowie k der kritische

Index.
Dann gibt es eine optimale Lösung $\mathbf{x}^*$ von (R) mit folgenden Eigenschaften:

- Gilt für ein $j \in \{1, 2, \ldots, k-1\}$

$$f_{LP} - f_G \leq c_j - \frac{a_j\, c_k}{a_k} \ ,$$

  dann ist $x_j^* = 1$.

- Gilt für ein $j \in \{k+1, k+2, \ldots, n\}$

$$f_{LP} - f_G < \frac{a_j\, c_k}{a_k} - c_j \ ,$$

  dann ist $x_j^* = 0$.

Bemerkung: Theorem 1 reduziert Problem (R) auf ein *Kernproblem.*

Beispiel 10: Reduktion des Rucksackproblems ✏

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

Kapitel 2

Metaheuristiken

2.1 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
→ 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** ⇒ Erzeuge Nachbarn zufällig.
⇒ Ersetze Zeile 2 im Algorithmus „Iterative Verbesserung“ durch

2^*: 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 Zfw.verbesserung erzeugt wurde, ohne den Zfw. weiter zu verbessern.

Betrachtet wird

$$\boxed{\begin{array}{c} f(\mathbf{x}) \to \min! \quad (\max!) \\ \text{u.d.N.} \\ \mathbf{x} \in S \subseteq \{0,1\}^n \end{array}} \tag{2.1}$$

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

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

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

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

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

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

Beispiel 1: Nachbarschaftssuche ✏

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.
→ Greedy-Algorithmus liefert nicht notwendig eine optimale Lösung.

Sei G eine 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.

2.2 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

2.3 Tabu-Suche

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

$\rightarrow$ 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)

Beispiel 2: Tabu-Liste ✏

2.4 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 |

Beispiel: Binäre Codierung ✏

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** |

$\to$

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 |

$\to$

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

⇒ 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

Beispiel 3: Genetische Operatoren ✏

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: $N^1 \subseteq N^2 \subseteq \ldots \subseteq N^k$

Kapitel 3

Dynamische Optimierung

→ Es werden Probleme betrachtet, die in einzelne „Stufen“ zerlegt werden können, so dass die Gesamtoptimierung durch eine „stufenweise Optimierung“ ersetzt werden kann.

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

3.1 Einführungsbeispiele

3.1.1 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 $(j = 1, 2, \ldots, n)$

Nebenbedingungen und Kostenbetrachtung ✏

Optimierungsproblem:

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

u.d.N.

$$\begin{aligned} x_{j+1} &= x_j + u_j - r_j, && j = 1, 2, \ldots, n \\ x_1 &= x_{n+1} = 0 \\ x_j &\geq 0, && j = 2, 3, \ldots, n \\ u_j &\geq 0, && j = 1, 2, \ldots, n \end{aligned} \tag{3.1}$$

Bemerkung:

$$x_1 = x_{n+1} = 0 \text{ und } (3.1)$$

$\Rightarrow$ 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:

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

3.1.2 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.

$$\begin{aligned} \sum_{j=1}^{n} a_j u_j &\leq V \\ u_1, u_2, \ldots, u_n &\in \{0, 1\} \end{aligned}$$

→ 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 $j,\ j+1, \ldots, n$

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

Umformuliertes Optimierungsproblem:

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

u.d.N.

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

3.2 Problemstellung

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

Zustandsvariable x_j:

→ beschreibt Zustand des Systems zu Beginn der Periode j (bzw. am Ende der Periode $j-1$)
→ $x_1 := x_a$ - vorgegebener Anfangszustand des Systems

Entscheidungsvariable u_j:

→ 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)

Illustration ✏

Optimierungsproblem:

$$\sum_{j=1}^{n} g_j(x_j, u_j) \;\to \min!$$

u.d.N.

$$\begin{aligned} & x_{j+1} = f_j(x_j, u_j), && j = 1, 2, \ldots, n \\ & x_1 = x_a, \\ & x_{j+1} \in X_{j+1}, && j = 1, 2, \ldots, n \\ & u_j \in U_j(x_j), && j = 1, 2, \ldots, n \end{aligned} \tag{3.2}$$

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 (3.2) genügt, heißt *zulässig*.

3.3 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 $1, \ldots, j-1$.

Illustration ✏

Bellmansche Funktionalgleichungen:

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

$$\boxed{\begin{aligned} v_j^*(x_j) &= g_j(x_j, u_j^*) + v_{j+1}^*(x_{j+1}^*) \\ &= \min_{u_j \in U_j(x_j)} \left\{ g_j(x_j, u_j) + v_{j+1}^*[f_j(x_j, u_j)] \right\} \\ x_j &\in X_j \end{aligned}} \tag{3.3}$$

   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.

2. 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 (3.3) 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) $\max_{1 \le j \le n} \big\{ g_j(x_j, u_j) \big\} \rightarrow \min!$

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

3.4 Bellmansche Funktionalgleichungsmethode

$\Rightarrow$ sukzessive Auswertung von (3.3) 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 $P_1(x^*_1 = x_a)$

Illustration: Bestimmung von $z^*_j(x_j)$ ✏

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 3.1.).

3.5 Beispiele und Anwendungen

3.5.1 Das binäre Rucksackproblem

Annahme: V, a_j, c_j - ganzzahlig

$$\begin{aligned} g_j(x_j, u_j) &= c_j u_j, & j &= 1, 2, \ldots, n \\ f_j(x_j, u_j) &= x_j - a_j u_j, & j &= 1, 2, \ldots, n \\ X_{j+1} &= \{0, 1, \ldots, V\} \end{aligned}$$

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

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:

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

Beispiel 1: Anwendung der dynamischen Optimierung zur Lösung des binären Rucksackproblems ✏

Illustration: Zustands- und Steuerbereiche ✏

3.5.2 Bestimmung eines kürzesten (längsten) Weges in einem Graphen

Illustration: Kürzeste-Wege-Problem ✏

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, $\quad 2 \leq j \leq n$

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

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

$v_j^*(x_j)$ - 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:

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

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

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

Beispiel 2: Kürzeste-Wege-Problem ✏

3.5.3 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:

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

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:

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

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

Anwendung der dynamischen Optimierung zur Lösung des Problems ✏

3.5.4 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:

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

u.d.N.

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

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

BFGL:

$$v_3^*(x_3) = a_3x_3 \ \ (= a_3u_3)$$

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

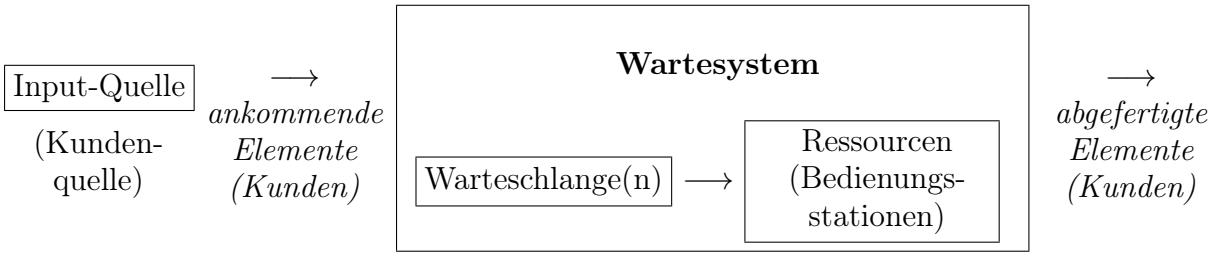
Anwendung der dynamischen Optimierung zur Lösung des Problems ✏

Kapitel 4

Warteschlangen

4.1 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

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

Annahmen über Zufallsgrößen:

Z_n unabhängig, identisch verteilt ⇒ Zufallsgröße Z (*Zwischenankunftszeit*)
S_n unabhängig, identisch verteilt ⇒ Zufallsgröße S (*Bedienungszeit*)
Z und S unabhängig voneinander

Illustration: Realisation von $\mathcal{L}(t)$ ✏

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 (identischen) 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 | M | 1 $\quad \rightarrow Z, S$ 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 | M | s | | K $\quad \rightarrow Z, S$ exponentialverteilte Zufallsgrößen, s Bedienungsschalter, K relevante Input-Elemente

4.2 Das System M|M|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

Eigenschaft von Punktprozessen mit exponentialverteilten Abständen der Ereignisse:

$$P(\text{Ereignis in } (t, t+\Delta)) = \lambda\Delta + o(\Delta)$$
$$P(\text{mehr als ein Ereignis in } (t, t+\Delta) = o(\Delta)$$

wobei λ - Parameter der Exponentialverteilung und

$$\lim_{\Delta \to 0} \frac{o(\Delta)}{\Delta} = 0$$

Ereignis: Eintreffen oder Abfertigen eines Kunden

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

Zur Anzahl der Kunden im System

Zustandswahrscheinlichkeit:

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

Betrachtet werden die Zeitpunkte t und $t + \Delta$.
$\mathcal{L}(t + \Delta) = n > 0$ kann wie folgt entstanden sein:

1. $$\mathcal{L}(t) = n - 1, \quad \mathcal{L}(t + \Delta) = n$$

   $\Rightarrow$ Wahrscheinlichkeit ist:
   $$\alpha\Delta P_{n-1}(t) + o(\Delta)$$

2. $$\mathcal{L}(t) = n + 1, \quad \mathcal{L}(t + \Delta) = n$$

   $\Rightarrow$ Wahrscheinlichkeit ist:
   $$\beta\Delta P_{n+1}(t) + o(\Delta)$$

3. $$\mathcal{L}(t) = n, \quad \mathcal{L}(t + \Delta) = n$$

   $\Rightarrow$ Wahrscheinlichkeit ist:
   $$(1 - \alpha\Delta - \beta\Delta)P_n(t) + o(\Delta)$$

4. in $(t,\ t+\Delta)$ erfolgt mehr als eine Ankunft und/oder Bedienung
   $\Rightarrow$ Wahrscheinlichkeit ist
$$o(\Delta)$$

$n = 0$ - Bedienung nicht möglich

$$P_0(t+\Delta) = \beta\Delta P_1(t) + (1-\alpha\Delta)P_0(t) + o(\Delta) \tag{4.1}$$

$n > 0$ - Fälle (1) - (4)

$$P_n(t+\Delta) = \alpha\Delta P_{n-1}(t) + \beta\Delta P_{n+1}(t) + (1-\alpha\Delta-\beta\Delta)P_n(t) + o(\Delta) \tag{4.2}$$

$\Rightarrow$ Dividiere (4.1),(4.2) durch Δ

$$\lim_{\Delta\to 0}\frac{P_0(t+\Delta)-P_0(t)}{\Delta} = \frac{dP_0(t)}{dt} = \beta P_1(t) - \alpha P_0(t)$$
$$\lim_{\Delta\to 0}\frac{P_n(t+\Delta)-P_n(t)}{\Delta} = \frac{dP_n(t)}{dt} = \alpha P_{n-1}(t) + \beta P_{n+1}(t) - (\alpha+\beta)P_n(t)$$

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 $\varrho = \dfrac{\alpha}{\beta}$

Beispiel 1: M|M|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

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

$$L = \frac{\alpha}{\beta - \alpha} \tag{4.3}$$

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

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

Mittlere Warte- und Verweilzeit der Kunden

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

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

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

Illustration: F_q ✏

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

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

analog:

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

$$W = E(\mathcal{W}) = W^q + \frac{1}{\beta} = \frac{1}{\beta - \alpha} \tag{4.4}$$

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

Aus Beziehung (4.3) und (4.4) folgt:

$$\boxed{\begin{array}{c} L = \alpha \cdot W \\ \text{analog: } L^q = \alpha \cdot W^q \\ \textbf{Little's Formel} \end{array}}$$

Beispiel 2: M|M|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?

4.3 Das System M|M|1|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|M|1|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|M|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?

4.4 Das System M|M|s

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

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

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|M|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)

4.5 Das System M|M|s|K mit endlichem Warteraum

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

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

mit

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

mittlere Anzahl der Kunden in der Warteschlange:

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

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|M|s|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;

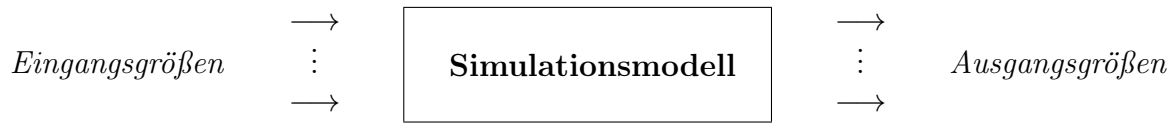
(c) ein Anrufer das Besetztzeichen hört?

Kapitel 5

Simulation

5.1 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

→ 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)

Beispiel 1: Produktion ✏

In einem Produktionsbetrieb seien 3 Aufträge $A_1,\ A_2,\ A_3$ auf 2 Maschinen $M_1,\ M_2$ zu bearbeiten. Alle Aufträge müssen zuerst auf M_1 und dann auf M_2 bearbeitet werden. Auf allen Maschinen ist die gleiche Auftragsreihenfolge zu wählen.

	Bearbeitungszeit [h]		
	A_1	A_2	A_3
M_1	3	1	2
M_2	2	4	1

Es gibt ein Eingangslager (vor M_1), ein Zwischenlager (zwischen M_1 und M_2) und ein Ausgangslager (nach M_2). Die gesuchte Größe ist die *Gesamtdurchlaufzeit* (Bearbeitungsende des letzten Auftrages auf M_2).

Beispiel 2: Materialwirtschaft

An einem Materialausgabeschalter werden Ersatzteile nach der FCFS Regel ausgegeben.

Verwendete Größen:

- i - Kundenindex
- t_i - Zeitpunkt der Ankunft des i-ten Kunden
- a_i - Zeit zwischen Ankunft des i-ten und $(i-1)$-ten Kunden
- b_i - Bedienungsdauer des i-ten Kunden
- e_i - Zeitpunkt des Bedienungsendes des i-ten Kunden
- w_i - Wartezeit des i-ten Kunden
- I - Anzahl der Kunden in einem Simulationslauf

- t_{ges} - gesamte simulierte Zeit, d.h.
  $t_{ges} = \sum_{i=1}^{I} a_i$
- b_{ges} - gesamte Bedienungszeit, d.h.
  $b_{ges} = \sum_{i=1}^{I} b_i$
- w_{ges} - gesamte Wartezeit, d.h.
  $w_{ges} = \sum_{i=1}^{I} w_i$

Ausgangsgrößen:

- ϱ - Auslastung der Bedienungsperson, d.h.
  $\varrho = \frac{b_{ges}}{t_{ges}}$
- $\overline{w}$ - mittlere Wartezeit pro Kunde, d.h.
  $\overline{w} = \frac{w_{ges}}{I}$
- $\overline{s}$ - mittlere Schlangenlänge, d.h.
  $\overline{s} = \frac{w_{ges}}{t_{ges}}$

Simulation des Warteschlangenproblems:

1. Setze die Anfangsbedingungen:
   $t_0 := e_0 := 0;\ i := 1;$
2. Erzeuge die Zwischenankunftszeit a_i des i-ten Kunden;
3. Ermittle den Zeitpunkt t_i der Ankunft des i-ten Kunden: $t_i := t_{i-1} + a_i$;
4. Erzeuge die Bedienungsdauer b_i des i-ten Kunden;
5. **IF** $e_{i-1} \leq t_i$ **THEN**
   BEGIN $w_i := 0,\ e_i := t_i + b_i$ **END**
   ELSE
   BEGIN $w_i := e_{i-1} - t_i;\ e_i := e_{i-1} + b_i$ **END**;
6. **IF** $i < I$ **THEN**
   BEGIN $i := i + 1$; gehe zu Schritt 2; **END**;
7. Berechne die Werte $t_{ges},\ b_{ges},\ w_{ges}$;
8. Ermittle die gesuchten Größen $\varrho,\ \overline{w},\ \overline{s}$; **STOP**.

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): siehe Abschnitt 5.2
- *Erstellung eines Computerprogramms*
  (auch Nutzung kommerzieller Simulations-Software, siehe Abschnitt 5.3)
- *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*

5.2 Erzeugung von Zufallszahlen

(a) Erzeugung von $(0,1)$-gleichverteilten Zufallszahlen

linearer Kongruenzgenerator

$$x_i = (a \cdot x_{i-1} + c) \mod m \qquad \Rightarrow \qquad x_i \in [0, m)$$

$$u_i = \frac{x_i}{m}$$

$\Rightarrow$ $u_1, u_2, u_3, \ldots$ Folge $(0,1)$-gleichverteilter Zufallszahlen

oft: $c = 0 \rightarrow$ multiplikativer Kongruenzgenerator
z.B.

$$m = 2^{31} - 1, \qquad a = 950706376$$

(b) Zufallszahlen beliebiger Verteilung

Erzeugung aus $(0,1)$-gleichverteilten Zufallszahlen mittels
inverser Tansformationsmethode

Sei X Zufallsgröße mit $P(X \leq t) = F(t)$ und z eine $(0,1)$-gleichverteilte Zufallszahl
$\Rightarrow$ Realisation der Zufallsgröße X:

$$t = F^{-1}(z)$$

Illustration ✏

Inverse Transformationsmethode auch für stetige Zufallsgrößen geeignet:

z.B. X ist exponentialverteilt

$$\Rightarrow \quad F(t) = 1 - e^{-\lambda t}, \qquad z = 1 - e^{-\lambda t}$$

$$\Rightarrow \; e^{-\lambda t} = 1 - z$$

$$\Rightarrow \; -\lambda t = \ln(1 - z)$$

$$\Rightarrow \; t = -\frac{1}{\lambda} \cdot \ln(1 - z)$$

Vereinfachung:

$$t = -\frac{1}{\lambda} \cdot \ln(z)$$

BEISPIEL 3: ZUFALLSZAHLEN ✏

Sei X eine diskrete Zufallsgröße mit

$$P_1 = P(X = x_1) = 0,14;\ P_2 = P(X = x_2) = 0,17;\ P_3 = P(X = x_3) = 0,22;$$
$$P_4 = P(X = x_4) = 0,16;\ P_5 = P(X = x_5) = 0,12;\ P_6 = P(X = x_6) = 0,11;$$
$$P_7 = P(X = x_7) = 0,08$$

5.3 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

  → *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