Vorlesung 11: Modellierung A

Grundlagen Operations Research

Modellierung A

Vom Problem zum Modell

Wie lernt man Modellierung?

- **Üben, üben, üben!**
- **Standardmodelle** verstehen und ihre Idee auf eigene Probleme übertragen.
- Verstehen: die Nebenbedingungen beschreiben nur den **zulässigen Lösungsraum**.
- Die (optimale) Lösung findet dann ein **Standardalgorithmus**.

Die richtigen Fragen

Vor jeder Formulierung:

- Worüber ist zu **entscheiden**?
- Was ist **entscheidungsrelevant**?
- Welche **Informationen** sind gegeben und relevant?
- Welche **Variablen** und von welchem Typ?
- Welche **Parameter** (Daten) werden gebraucht?

Der Modellierungsprozess

1. **Problem**: die reale Fragestellung.
2. **Mathematisches Modell**: Variablen, Zielfunktion, Nebenbedingungen.
3. **Optimierung**: ein Standardalgorithmus löst das Modell.
4. **Ergebnis**: Interpretation und Rückübersetzung.

Problemtypen: **LP**, **MIP** (gemischt-ganzzahlig), **NLP** (nichtlinear), **MINLP**.

Werkzeuge

- **Algorithmen**: Simplex (LP), Branch & Bound (MIP), Netzwerkfluss-Verfahren, Dekomposition, Outer Approximation (MINLP), (Meta-)Heuristiken.
- **Modellierungssprachen**: früher AMPL, LINGO, GAMS, hier **Julia mit JuMP**.

Eine Modellierungssprache trennt das **Modell** sauber vom **Solver**.

Klassische Modelle

Mischungsproblem

Ein Fleischer mischt Hackfleisch mit **höchstens 25 % Fett**.

	Preis/kg	Fettgehalt
Rind (R)	5,75 €	15 %
Schwein (S)	4,00 €	30 %

Gesucht: das **kostenminimale** Mischungsverhältnis für 1 kg.

Frage: **Worüber ist zu entscheiden, und welche Restriktionen sehen Sie?**

Mischungsproblem: Modell

$$
\begin{aligned}
\min & 5,75\,R + 4,00\,S \\
& R + S = 1 \qquad\qquad (1\ \text{kg Mischung}) \\
& 0,15\,R + 0,30\,S \le 0,25 (\text{max. } 25\,\%\ \text{Fett}) \\
& R, S \ge 0
\end{aligned}
$$

```
using JuMP, HiGHS
modell = Model(HiGHS.Optimizer); set_silent(modell)
@variable(modell, R >= 0); @variable(modell, S >= 0)
@objective(modell, Min, 5.75R + 4S)
@constraint(modell, R + S == 1)
@constraint(modell, 0.15R + 0.30S <= 0.25)
optimize!(modell)
println("R = ", round(value(R), digits=3), ", S = ", round(value(S),
digits=3),
        ", Kosten = ", round(objective_value(modell), digits=2))
```

```
R = 0.333, S = 0.667, Kosten = 4.58
```

Produktionsprogrammplanung

Produkt	Deckungsbeitrag	Anlage 1	Anlage 2	Höchstmenge
A	60	3	1	100
B	28	1	4	150
Kapazität		500	500	

$$
\begin{aligned}
\max & 60A + 28B \\
& 3A + B \le 500, \quad A + 4B \le 500 \\
& A \le 100, \quad B \le 150, \quad A, B \ge 0
\end{aligned}
$$

Lösung und Schattenpreise

Frage: **Was wäre mehr Kapazität wert?**

```
using JuMP, HiGHS
modell = Model(HiGHS.Optimizer); set_silent(modell)
```

```
@variable(modell, 0 <= A <= 100); @variable(modell, 0 <= B <= 150)
@objective(modell, Max, 60A + 28B)
@constraint(modell, anlage1, 3A + B <= 500)
@constraint(modell, anlage2, A + 4B <= 500)
optimize!(modell)
println("A = ", round(Int, value(A)), ", B = ", round(Int, value(B)),
        ", F* = ", round(Int, objective_value(modell)))
println("Schattenpreise: Anlage 1 = ", round(Int, shadow_price(anlage1)),
        ", Anlage 2 = ", round(Int, shadow_price(anlage2)))
```

```
A = 100, B = 100, F* = 8800
Schattenpreise: Anlage 1 = 0, Anlage 2 = 7
```

Anlage 1 hat Puffer (Preis **0**); Anlage 2 ist bis **7 GE/ZE** eine Erweiterung wert (Vorlesung 4!).

Lohnt sich Produkt C?

Produkt C würde **je 2 ZE** auf beiden Anlagen beanspruchen.

Frage: **Ab welchem Deckungsbeitrag gehört C ins Programm?**

Opportunitätskosten über die Schattenpreise (Vorlesung 3):

$$2 \cdot 0 + 2 \cdot 7 = 14$$

! Ergebnis

Erst mit einem Deckungsbeitrag **über 14** verdrängt Produkt C die bestehende Lösung.

Personalplanung

Ein Supermarkt (24 h) braucht je 4-Stunden-**Periode** eine Mindestzahl an Kassierern; eine Schicht dauert **8 h** (zwei Perioden), Periode 1 folgt auf 6.

Periode	1	2	3	4	5	6
Uhrzeit	3–7	7–11	11–15	15–19	19–23	23–3
Bedarf b_t	7	20	14	20	10	5

Frage: **Was ist die Entscheidungsvariable?**

X_t: Anzahl der Kassierer, die zu **Beginn von Periode** t starten (und zwei Perioden arbeiten).

Personalplanung: Modell

$$\begin{aligned}
&\min \sum_{t=1}^{6} X_t \\
&X_t + X_{t-1} \geq b_t \qquad \text{für alle Perioden } t \text{(zyklisch)} \\
&X_t \geq 0 \text{ und ganzzahlig}
\end{aligned}$$

```
b = [7, 20, 14, 20, 10, 5]
modell = Model(HiGHS.Optimizer); set_silent(modell)
@variable(modell, X[1:6] >= 0, Int)
@objective(modell, Min, sum(X))
@constraint(modell, [t in 1:6], X[t] + X[t == 1 ? 6 : t-1] >= b[t])
optimize!(modell)
println("Kassierer gesamt: ", round(Int, objective_value(modell)),
        " mit X = ", round.(Int, value.(X)))
```

```
Kassierer gesamt: 45 mit X = [20, 0, 15, 5, 5, 0]
```

45 Kassierer genügen (eine von mehreren optimalen Aufteilungen).

Dynamische Losgrößenplanung

Das Problem

Ein Produkt, bekannter Bedarf je Periode. Wann und wie viel **produzieren**, um **Rüst-** und **Lagerkosten** zu minimieren?

- Produktion und Lagerung zu **Periodenbeginn**, keine Kapazitätsgrenze.
- **Keine** Fehlmengen; fester Planungshorizont.
- Feste **Rüstkosten** je Auftrag, **Lagerkosten** je eingelagerter Einheit.

Frage: **Worüber entscheiden wir hier eigentlich?**

Notation

Entschieden wird über einen **binären Schalter** (rüsten?) und eine **kontinuierliche Menge** (wie viel?); der Lagerbestand folgt daraus:

- $\mathcal{T}$: Perioden ($t = 1, \ldots, |\mathcal{T}|$); d_t: Nachfrage in Periode t.
- f: fixe Rüstkosten; h: Lagerhaltungskostensatz.
- $X_t \in \{0, 1\}$: wird in t gerüstet? Q_t: Produktionsmenge.
- I_t: Lagerbestand am Ende von t ($I_0 = 0$).

Das Modell

$$\begin{aligned}
&\min F = \sum_t (f\, X_t + h\, I_t) \\
&I_{t-1} + Q_t - I_t = d_t \qquad \forall t \quad \text{(Lagerbilanz)} \\
&Q_t \leq b\, X_t \qquad \forall t \quad \text{(nur mit Rüsten)} \\
&Q_t, I_t \geq 0, \quad X_t \in \{0, 1\} \forall t
\end{aligned}$$

Idee: Big-M als Wenn-dann-Schalter

b ist eine große Zahl (z. B. die Gesamtnachfrage). $X_t = 0$ erzwingt $Q_t = 0$; $X_t = 1$ gibt Q_t frei, so wird „produzieren **nur mit** Rüsten" linear.

Frage: **Was passiert, wenn wir $Q_t \le b\, X_t$ weglassen?**

Dann würde produziert, **ohne** je Rüstkosten zu zahlen: X_t bliebe einfach null.

Beispiel-Daten

$f = 500$, $h = 1$:

t	1	2	3	4	5	6	7	8	9	10	11	12
d_t	140	110	190	120	200	110	250	100	220	100	180	130
X_t^*	1	0	1	0	1	0	1	0	1	0	1	0
Q_t^*	250	0	310	0	310	0	350	0	320	0	310	0
I_t^*	110	0	120	0	110	0	100	0	100	0	130	0

! Ergebnis

$F^* = 3670$: gerüstet wird nur in **ungeraden** Perioden, jeweils für zwei Perioden vorproduziert (sichtbar am Lagerbestand I_t^*).

Kontrolle mit Julia

```
using JuMP, HiGHS

d = [140,110,190,120,200,110,250,100,220,100,180,130]
f, h = 500, 1
T = 1:length(d); b = sum(d)

modell = Model(HiGHS.Optimizer); set_silent(modell)
@variable(modell, Q[T] >= 0); @variable(modell, I[T] >= 0); @variable(modell,
X[T], Bin)
@objective(modell, Min, sum(f*X[t] + h*I[t] for t in T))
@constraint(modell, [t in T], (t == 1 ? 0 : I[t-1]) + Q[t] - I[t] == d[t])
@constraint(modell, [t in T], Q[t] <= b * X[t])

optimize!(modell)
println("F* = ", round(Int, objective_value(modell)),
        " mit ", round(Int, sum(value.(X))), " Rüstvorgängen")
```

```
F* = 3670 mit 6 Rüstvorgängen
```

Zusammenfassung

Das Wichtigste auf einen Blick

- Modellierung heißt: **entscheiden**, was Variablen, Ziel und Restriktionen sind, geübt an **Standardmodellen**.
- **Mischung**, **Produktionsprogramm** und **Personalplanung** sind klassische LP/MIP-Bausteine.
- Die **dynamische Losgrößenplanung** koppelt kontinuierliche Mengen an binäre Rüstentscheidungen ($Q_t \leq b\, X_t$).
- Mit **JuMP** wird jedes dieser Modelle direkt lösbar.

Literatur

Literaturverzeichnis

Bibliografie