

Vorlesung 11: Modellierung A

Grundlagen Operations Research

Prof. Dr. Knut Haase

Modellierung A

Vom Problem zum Modell

Wie lernt man Modellierung?

Wie lernt man Modellierung?

- Üben, üben, üben!

Wie lernt man Modellierung?

- Üben, üben, üben!
- **Standardmodelle** verstehen und ihre Idee auf eigene Probleme übertragen.

Wie lernt man Modellierung?

- Üben, üben, üben!
- **Standardmodelle** verstehen und ihre Idee auf eigene Probleme übertragen.
- Verstehen: die Nebenbedingungen beschreiben nur den **zulässigen Lösungsraum**.

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:

Die richtigen Fragen

Vor jeder Formulierung:

- Worüber ist zu **entscheiden**?

Die richtigen Fragen

Vor jeder Formulierung:

- Worüber ist zu **entscheiden**?
- Was ist **entscheidungsrelevant**?

Die richtigen Fragen

Vor jeder Formulierung:

- Worüber ist zu **entscheiden**?
- Was ist **entscheidungsrelevant**?
- Welche **Informationen** sind gegeben und relevant?

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?

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

Der Modellierungsprozess

1. **Problem:** die reale Fragestellung.

Der Modellierungsprozess

1. **Problem:** die reale Fragestellung.
2. **Mathematisches Modell:** Variablen, Zielfunktion, Nebenbedingungen.

Der Modellierungsprozess

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

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.

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**.

Optimierungsalgorithmen

Optimierungsalgorithmen

- Simplex-Algorithmus für LP.

Optimierungsalgorithmen

- Simplex-Algorithmus für LP.
- Graphentheoretische Verfahren (Netzwerkfluss, kürzester Weg).

Optimierungsalgorithmen

- Simplex-Algorithmus für LP.
- Graphentheoretische Verfahren (Netzwerkfluss, kürzester Weg).
- Branch & Bound für MIP.

Optimierungsalgorithmen

- Simplex-Algorithmus für LP.
- Graphentheoretische Verfahren (Netzwerkfluss, kürzester Weg).
- Branch & Bound für MIP.
- Outer Approximation für (konvexe) MINLP.

Optimierungsalgorithmen

- Simplex-Algorithmus für LP.
- Graphentheoretische Verfahren (Netzwerkfluss, kürzester Weg).
- Branch & Bound für MIP.
- Outer Approximation für (konvexe) MINLP.
- Dekompositionsverfahren (Lagrange-Relaxation, Benders-Dekomposition, Spaltengenerierung).

Heuristiken und Simulation

Heuristiken und Simulation

- Mathematische Heuristiken (z. B. Fix-and-Optimize).

Heuristiken und Simulation

- **Mathematische Heuristiken** (z. B. Fix-and-Optimize).
- **Heuristiken** (Greedy, Konstruktionsverfahren, n-opt), insbesondere **Metaheuristiken** (Tabu-Suche, Simulated Annealing, genetische Algorithmen, Ameisenalgorithmen).

Heuristiken und Simulation

- **Mathematische Heuristiken** (z. B. Fix-and-Optimize).
- **Heuristiken** (Greedy, Konstruktionsverfahren, n-opt), insbesondere **Metaheuristiken** (Tabu-Suche, Simulated Annealing, genetische Algorithmen, Ameisenalgorithmen).
- **Simulationsgestützte Optimierung** (stochastische Probleme).

Modellierungssprachen

Modellierungssprachen

- Einfacher Zugriff auf aktuelle **Solver**.

Modellierungssprachen

- Einfacher Zugriff auf aktuelle **Solver**.
- Einfache **Implementierung** von Modellen.

Modellierungssprachen

- Einfacher Zugriff auf aktuelle **Solver**.
- Einfache **Implementierung** von Modellen.
- Leichte **Anpassung** von Lösungsansätzen.

Modellierungssprachen

- Einfacher Zugriff auf aktuelle **Solver**.
- Einfache **Implementierung** von Modellen.
- Leichte **Anpassung** von Lösungsansätzen.
- Software: früher AMPL, LINGO, GAMS, hier **Julia mit JuMP**.

Modellierungssprachen

- Einfacher Zugriff auf aktuelle **Solver**.
- Einfache **Implementierung** von Modellen.
- Leichte **Anpassung** von Lösungsansätzen.
- Software: 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.

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

$$\min 5,75 R + 4,00 S$$

$$R + S = 1 \quad (1 \text{ kg Mischung})$$

$$0,15 R + 0,30 S \leq 0,25 \quad (\text{max. 25 \% Fett})$$

$$R, S \geq 0$$

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

Mischungsproblem: Modell

$$\min 5,75 R + 4,00 S$$

$$R + S = 1 \quad (1 \text{ kg Mischung})$$

$$0,15 R + 0,30 S \leq 0,25 \quad (\text{max. 25 \% Fett})$$

$$R, S \geq 0$$

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

R = 0.333, S = 0.667, Kosten = 4.58

Produktionsprogrammplanung

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

$$\max 60A + 28B$$

$$3A + B \leq 500, \quad A + 4B \leq 500$$

$$A \leq 100, \quad B \leq 150, \quad A, B \geq 0$$

Lösung und Schattenpreise

Frage: Was wäre mehr Kapazität wert?

```
1 using JuMP, HiGHS
2 modell = Model(HiGHS.Optimizer); set_silent(modell)
3 @variable(modell, 0 <= A <= 100); @variable(modell, 0 <= B <= 150)
4 @objective(modell, Max, 60A + 28B)
5 @constraint(modell, anlage1, 3A + B <= 500)
6 @constraint(modell, anlage2, A + 4B <= 500)
7 optimize!(modell)
8 println("A = ", round{Int, value(A)}, ", B = ", round{Int, value(B)},
9         ", F* = ", round{Int, objective_value(modell)})
10 println("Schattenpreise: Anlage 1 = ", round{Int, shadow_price(anlage1)},
11         ", Anlage 2 = ", round{Int, shadow_price(anlage2)})
```

Lösung und Schattenpreise

Frage: Was wäre mehr Kapazität wert?

```
1 using JuMP, HiGHS
2 modell = Model(HiGHS.Optimizer); set_silent(modell)
3 @variable(modell, 0 <= A <= 100); @variable(modell, 0 <= B <= 150)
4 @objective(modell, Max, 60A + 28B)
5 @constraint(modell, anlage1, 3A + B <= 500)
6 @constraint(modell, anlage2, A + 4B <= 500)
7 optimize!(modell)
8 println("A = ", round{Int, value(A)}, ", B = ", round{Int, value(B)},
9         ", F* = ", round{Int, objective_value(modell)})
10 println("Schattenpreise: Anlage 1 = ", round{Int, shadow_price(anlage1)),
11         ", Anlage 2 = ", round{Int, shadow_price(anlage2)})
```

A = 100, B = 100, F* = 8800

Schattenpreise: Anlage 1 = 0, Anlage 2 = 7

Schattenpreise deuten

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

Schattenpreise deuten

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

Schattenpreis = Simplex-Multiplikator

Der Schattenpreis einer Restriktion ist ihr **Simplex-Multiplikator** π_i aus Vorlesung 3 (im Optimaltableau unter der Schlupfvariablen Y_i). Vorlesung 4 nennt dieselbe Größe **Schattenpreis**, dort als Dualvariable W_i .

Lohnt sich Produkt C?

Produkt C würde je 2 ZE auf beiden Anlagen beanspruchen.

Lohnt sich Produkt C?

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

Frage: Ab welchem Deckungsbeitrag gehört C ins Programm?

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):

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

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

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?

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

$$\min \sum_{t=1}^6 X_t$$

$$X_t + X_{t-1} \geq b_t \quad \text{für alle Perioden } t \text{ (zyklisch)}$$

$$X_t \geq 0 \text{ und ganzzahlig}$$

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

Personalplanung: Modell

$$\min \sum_{t=1}^6 X_t$$

$$X_t + X_{t-1} \geq b_t \quad \text{für alle Perioden } t \text{ (zyklisch)}$$

$$X_t \geq 0 \text{ und ganzzahlig}$$

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

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

Personalplanung: Modell

$$\min \sum_{t=1}^6 X_t$$

$$X_t + X_{t-1} \geq b_t \quad \text{für alle Perioden } t \text{ (zyklisch)}$$

$$X_t \geq 0 \text{ und ganzzahlig}$$

```
1 b = [7, 20, 14, 20, 10, 5]
2 modell = Model(HiGHS.Optimizer); set_silent(modell)
3 @variable(modell, X[1:6] >= 0, Int)
4 @objective(modell, Min, sum(X))
5 @constraint(modell, [t in 1:6], X[t] + X[t == 1 ? 6 : t-1] >= b[t])
6 optimize!(modell)
7 println("Kassierer gesamt: ", round{Int, objective_value(modell)},
8         " 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).

Losgrößenplanung

Das Problem

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

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 Lieferfrist, keine Kapazitätsgrenze.

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 Lieferfrist, keine Kapazitätsgrenze.
- **Keine** Fehlmengen; fester Planungshorizont.

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 Lieferfrist, keine Kapazitätsgrenze.
- **Keine** Fehlmengen; fester Planungshorizont.
- Feste **Rüstkosten** je Auftrag, **Lagerkosten** je eingelagerter Einheit.

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 Lieferfrist, 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:

Notation

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

- $\mathcal{T}$: Perioden ($t = 1, \dots, |\mathcal{T}|$); d_t : Nachfrage in Periode t .

Notation

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

- $\mathcal{T}$: Perioden ($t = 1, \dots, |\mathcal{T}|$); d_t : Nachfrage in Periode t .
- f : fixe Rüstkosten; h : Lagerhaltungskostensatz.

Notation

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

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

Notation

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

- $\mathcal{T}$: Perioden ($t = 1, \dots, |\mathcal{T}|$); 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 ($I_0 = 0$).

Implizite und explizite Form

Zwei Schreibweisen für dasselbe Modell:

Implizite und explizite Form

Zwei Schreibweisen für dasselbe Modell:

- **Implizite Form:** Das Modell steht mit **Symbolen** und Indexmengen da, wie eben $\mathcal{T}$, d_t , f und h . Die Daten folgen getrennt.

Implizite und explizite Form

Zwei Schreibweisen für dasselbe Modell:

- **Implizite Form:** Das Modell steht mit **Symbolen** und Indexmengen da, wie eben $\mathcal{T}$, d_t , f und h . Die Daten folgen getrennt.
- **Explizite Form:** Alle **Zahlen** stehen ausgeschrieben im Modell, wie in den ersten Vorlesungen, etwa $X_1 + X_2 \leq 5$.

Implizite und explizite Form

Zwei Schreibweisen für dasselbe Modell:

- **Implizite Form:** Das Modell steht mit **Symbolen** und Indexmengen da, wie eben $\mathcal{T}$, d_t , f und h . Die Daten folgen getrennt.
- **Explizite Form:** Alle **Zahlen** stehen ausgeschrieben im Modell, wie in den ersten Vorlesungen, etwa $X_1 + X_2 \leq 5$.

Der Vorteil der impliziten Form: Ändern sich die Daten, bleibt das **Modell unverändert**.

Das Modell

$$\min F = \sum_t (f X_t + h I_t)$$

$$I_{t-1} + Q_t - I_t = d_t \quad \forall t \quad (\text{Lagerbilanz})$$

$$Q_t \leq b X_t \quad \forall t \quad (\text{nur mit Rüsten})$$

$$Q_t, I_t \geq 0, \quad X_t \in \{0, 1\} \quad \forall t$$

Das Modell

$$\min F = \sum_t (f X_t + h I_t)$$

$$I_{t-1} + Q_t - I_t = d_t \quad \forall t \quad (\text{Lagerbilanz})$$

$$Q_t \leq b X_t \quad \forall t \quad (\text{nur mit Rüsten})$$

$$Q_t, I_t \geq 0, \quad X_t \in \{0, 1\} \quad \forall t$$

 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.

Das Modell

$$\min F = \sum_t (f X_t + h I_t)$$

$$I_{t-1} + Q_t - I_t = d_t \quad \forall t \quad (\text{Lagerbilanz})$$

$$Q_t \leq b X_t \quad \forall t \quad (\text{nur mit Rüsten})$$

$$Q_t, I_t \geq 0, \quad X_t \in \{0, 1\} \quad \forall t$$

 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 $Q_t \leq b X_t$ weglassen?

Das Modell

$$\min F = \sum_t (f X_t + h I_t)$$

$$I_{t-1} + Q_t - I_t = d_t \quad \forall t \quad (\text{Lagerbilanz})$$

$$Q_t \leq b X_t \quad \forall t \quad (\text{nur mit Rüsten})$$

$$Q_t, I_t \geq 0, \quad X_t \in \{0, 1\} \quad \forall t$$

 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 $Q_t \leq b X_t$ weglassen?

Dann würde **ohne** Rüstkosten produziert: X_t bliebe null.

Beispiel-Daten

$f = 500$, $h = 1$, Perioden 1 bis 12:

t	1	2	3	4	5	6
d_t	140	110	190	120	200	110
t	7	8	9	10	11	12
d_t	250	100	220	100	180	130

Beispiel-Daten

$f = 500$, $h = 1$, Perioden 1 bis 12:

t	1	2	3	4	5	6
d_t	140	110	190	120	200	110
t	7	8	9	10	11	12
d_t	250	100	220	100	180	130

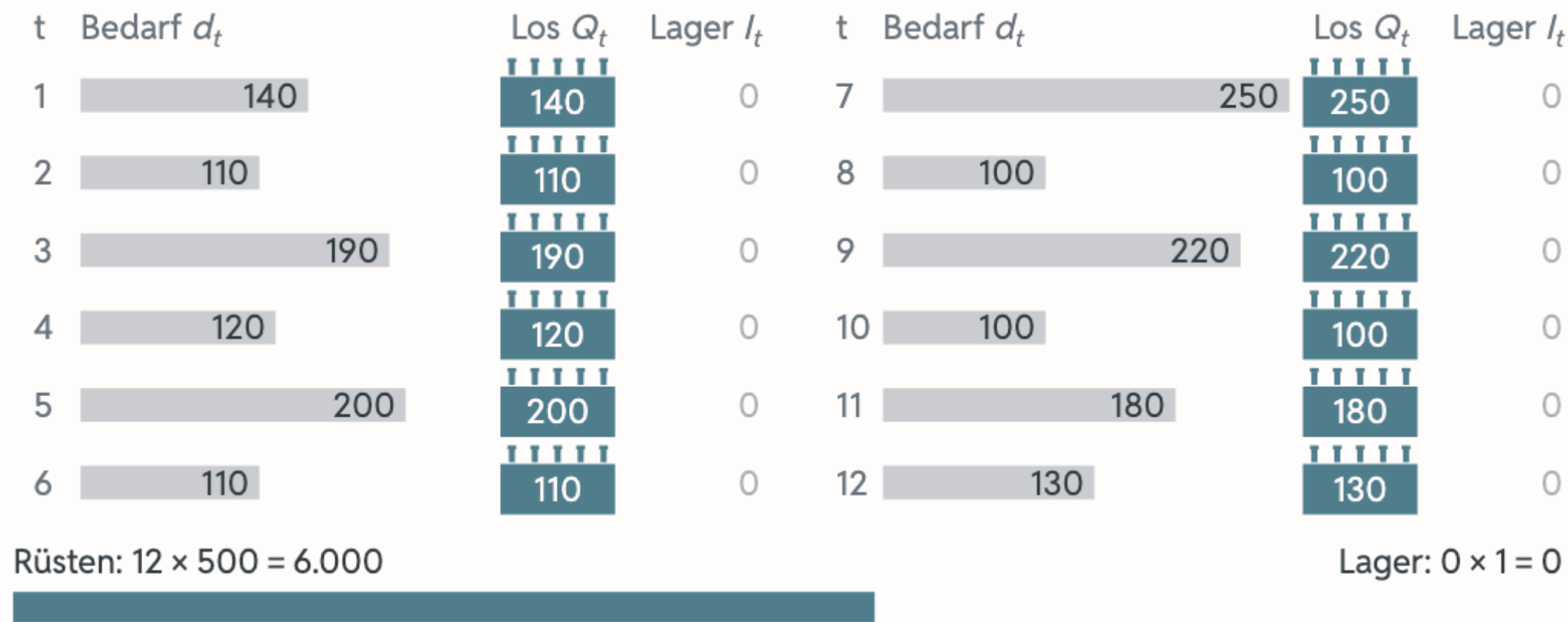


Gleich ausprobieren

Auf der nächsten Folie planen Sie mit diesen Daten selbst: In welchen Perioden rüsten Sie? In jeder Periode zu rüsten kostet 6000. Wie weit kommen Sie darunter?

Sie sind dran

Tippen Sie auf eine Periode, um dort zu rüsten oder nicht: Ein Los deckt den Bedarf bis zum nächsten Rüsten. Jedes Rüsten kostet 500, jede Einheit im Lager 1 je Periode.



Eigener Plan 6.000

Bereit: „Optimieren“
vergleicht den Plan
mit dem besten.

Optimieren

Reset

Neu

Optimum: Perioden 1 bis 6

Vergleichen Sie mit Ihrem Plan aus dem Spiel:

t	1	2	3	4	5	6
d_t	140	110	190	120	200	110
X_t^*	1	0	1	0	1	0
Q_t^*	250	0	310	0	310	0
I_t^*	110	0	120	0	110	0

Optimum: Perioden 7 bis 12

t	7	8	9	10	11	12
d_t	250	100	220	100	180	130
X_t^*	1	0	1	0	1	0
Q_t^*	350	0	320	0	310	0
I_t^*	100	0	100	0	130	0

Optimum: Perioden 7 bis 12

t	7	8	9	10	11	12
d_t	250	100	220	100	180	130
X_t^*	1	0	1	0	1	0
Q_t^*	350	0	320	0	310	0
I_t^*	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

```
1 using JuMP, HiGHS
2
3 d = [140,110,190,120,200,110,250,100,220,100,180,130]
4 f, h = 500, 1
5 T = 1:length(d); b = sum(d)
6
7 modell = Model(HiGHS.Optimizer); set_silent(modell)
8 @variable(modell, Q[T] >= 0); @variable(modell, I[T] >= 0)
9 @variable(modell, X[T], Bin)
10 @objective(modell, Min, sum(f*X[t] + h*I[t] for t in T))
11 @constraint(modell, [t in T], (t == 1 ? 0 : I[t-1]) + Q[t] - I[t] == d[t])
12 @constraint(modell, [t in T], Q[t] <= b * X[t])
13
14 optimize!(modell)
15 println("F* = ", round{Int, objective_value(modell)},
16         " mit ", round{Int, sum(value.(X))}, " Rüstvorgängen")
```

Kontrolle mit Julia

```
1 using JuMP, HiGHS
2
3 d = [140,110,190,120,200,110,250,100,220,100,180,130]
4 f, h = 500, 1
5 T = 1:length(d); b = sum(d)
6
7 modell = Model(HiGHS.Optimizer); set_silent(modell)
8 @variable(modell, Q[T] >= 0); @variable(modell, I[T] >= 0)
9 @variable(modell, X[T], Bin)
10 @objective(modell, Min, sum(f*X[t] + h*I[t] for t in T))
11 @constraint(modell, [t in T], (t == 1 ? 0 : I[t-1]) + Q[t] - I[t] == d[t])
12 @constraint(modell, [t in T], Q[t] <= b * X[t])
13
14 optimize!(modell)
15 println("F* = ", round{Int, objective_value(modell)},
16         " mit ", round{Int, sum(value.(X))}, " Rüstvorgängen")
```

F* = 3670 mit 6 Rüstvorgängen

Zusammenfassung

Das Wichtigste auf einen Blick

Das Wichtigste auf einen Blick

- Modellierung heißt: **entscheiden**, was Variablen, Ziel und Restriktionen sind, geübt an **Standardmodellen**.

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.

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 ($Q_t \leq b X_t$).

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

Domschke, Wolfgang, Andreas Drexl, Robert Klein, und Armin Scholl. 2015. *Einführung in Operations Research*. 9. Aufl. Springer Gabler. <https://doi.org/10.1007/978-3-662-48216-2>.