Einführung in die Optimierung

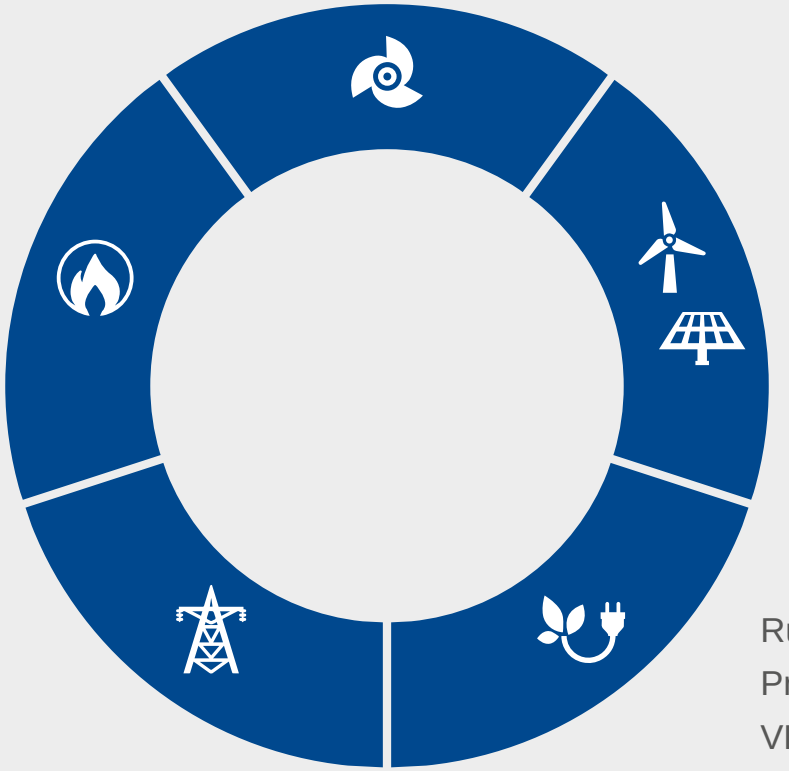
Ulrike Stöcker
Ulrike.stoecker@verbund.com

Wien 15.11.2024

VERBUND auf einen Blick

130 Wasserkraftwerke von VERBUND mit über 8.400 MW Leistung.

Rund **900** Kilometer ist das Erdgas-Hochdruckleitungsnetz der GCA lang.



Bis zu **1/4** der Gesamterzeugung soll bis 2030 aus Sonnen- und Windkraft kommen.

Rund **3.400** Kilometer Trassenlänge hat das überregionale Stromnetz der APG.

Rund **490.000** Privatkund:innen setzten 2023 auf VERBUND.

Nachhaltige Energiezukunft
98 % Erzeugung aus erneuerbaren Energien

Wasserkraft[1]
>92 % der Stromerzeugung
130 Wasserkraftwerke
30.509 GWh Strom

Windkraft
>4 % der Stromerzeugung
321 Windkraftanlagen
1.397 GWh Strom

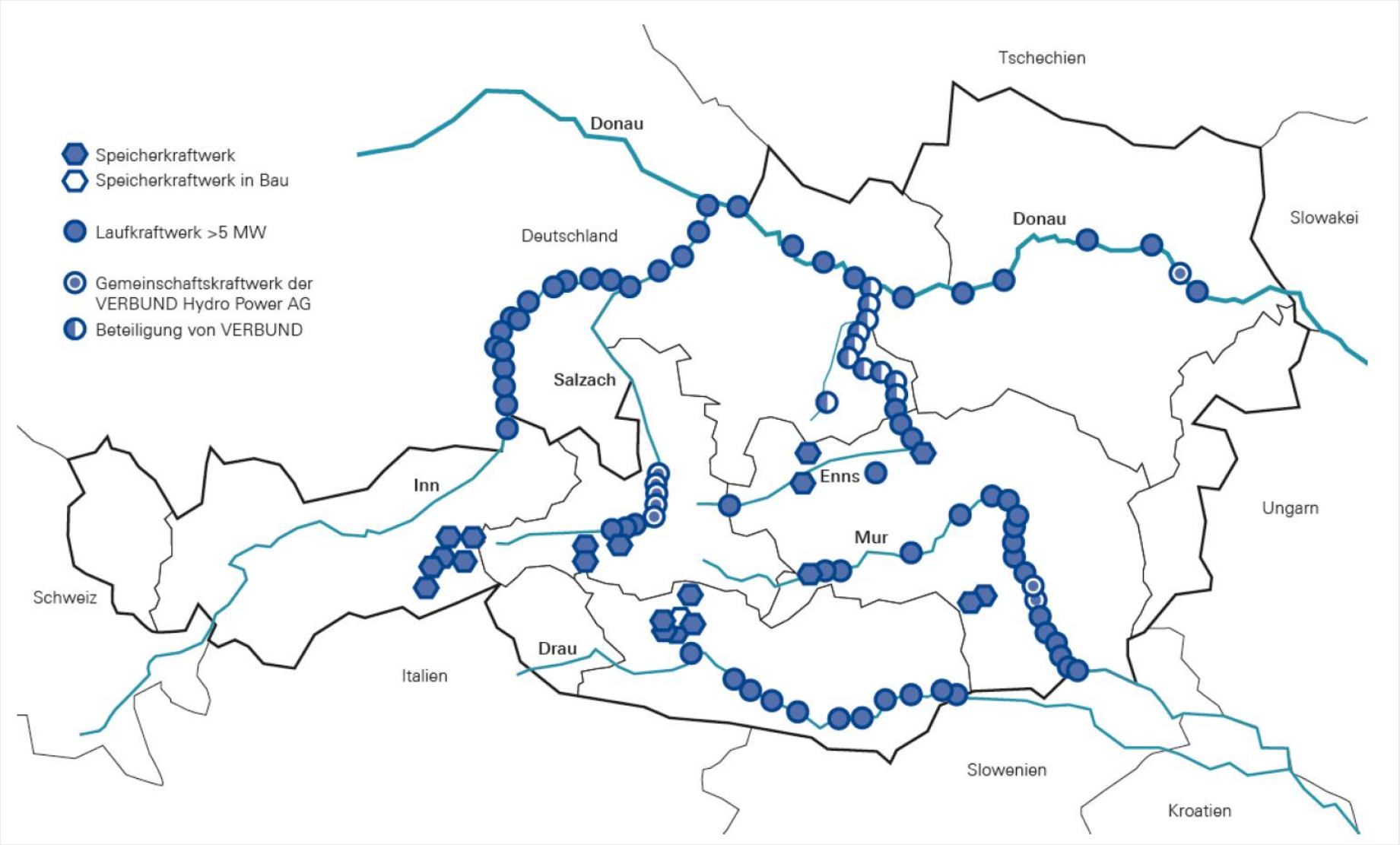
Sonnenkraft[2]
>1 % der Stromerzeugung
44 Photovoltaikparks
362 GWh Strom

Wärmekraft
>2 % der Stromerzeugung
2 Wärmekraftwerke
677 GWh Strom

Batteriespeicher
< 1 % der Stromerzeugung
15 Anlagen
32 GWh Strom

1 inklusive Bezugsrechte; ohne nicht-vollkonsolidierte Anlagen (Ashta 1&2 sowie Nussdorf)
2 ohne Leasing-/Contracting-Anlagen
Alle Werte IST Erzeugung 2023

Kraftwerkspark der VHP



Operation Research

Operation Research ist das wissenschaftliche Gebiet, welches sich mit modellbasierender Lösungshilfe für komplexe Managementprobleme auseinandersetzt. Es hat den Zweck, Einsichten in die Lösungen dieser Probleme zu erhalten. Als solches ist die Modellbildung, Analyse von Modellen und Interpretation von Modelllösungen Gegenstand dieser Disziplin.

Ein Modell ist eine Vereinfachte Darstellung der Wirklichkeit.
Es gibt immer wieder einen Trade-off zwischen Plausibilität und Durchführbarkeit.

In Operations Resarch kommen mathematische Modelle zum Einsatz.
Entscheidungen die getroffenen werden können strategischer (mittel- bis langfristig, ungenau) und taktischer (kurzfristig, genau) Natur sein.

Typsiche Problemstellungen

- Lagerhaltungsprobleme
- Warteschlangen-Problem
- Aufteilungsprobleme

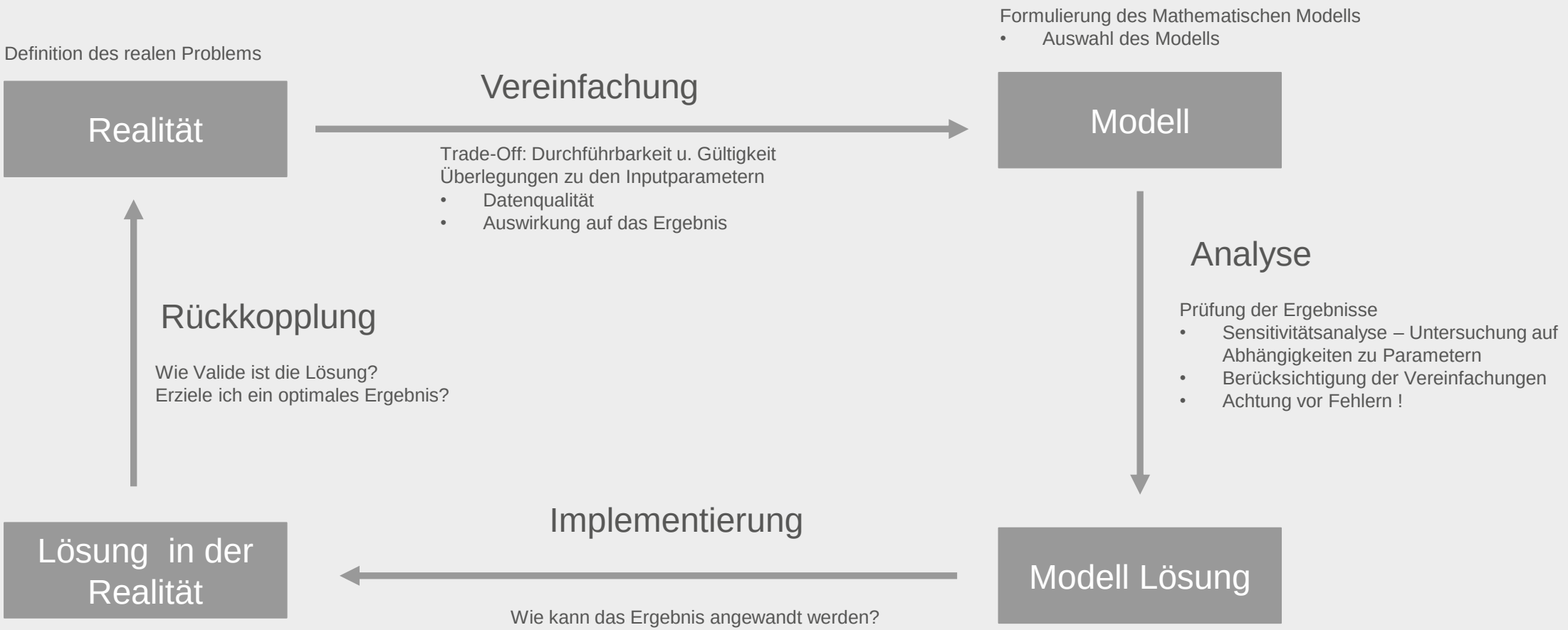
- Bsp: “Problem des Handlungsreisenden”:
  Ein Handlungsreisender muss hierbei die beste Route für einen Besuch in mehreren Städten planen, wobei keine Stadt mehrmals besucht werden soll. Um ein optimales Kosten-Nutzen-Verhältnis zu schaffen, soll die Strecke für diese Reise möglichst kurz sein und der Handlungsreisende am Ende wieder in seiner Ausgangsstadt eintreffen.

Verfahren:

- Simulationsverfahren
- Entscheidungsbaumverfahren
- Mathematische Optimierung
- Verfahren der Warteschlangentheorie
- Heuristische Verfahren…

V

OR Prozess


Definition des realen Problems
Realität
Vereinfachung
Trade-Off: Durchführbarkeit u. Gültigkeit
Überlegungen zu den Inputparametern
• Datenqualität
• Auswirkung auf das Ergebnis
Formulierung des Mathematischen Modells
• Auswahl des Modells
Modell
Analyse
Prüfung der Ergebnisse
• Sensitivitätsanalyse – Untersuchung auf Abhängigkeiten zu Parametern
• Berücksichtigung der Vereinfachungen
• Achtung vor Fehlern !
Modell Lösung
Implementierung
Wie kann das Ergebnis angewandt werden?
Lösung in der Realität
Rückkopplung
Wie Valide ist die Lösung?
Erziele ich ein optimales Ergebnis?

Modell-Überlegungen

Exakte vs. Approximative Lösung:
Exakte Lösung = beweisbare zulässige Lösung
Approximative Lösung = Zulässige Lösung einer präskriptiven Analyse (keine Garantie auf exaktes Optimum)

Deterministik – Stochastik:
Deterministisches math. Modell: alle Parameter Werte gelten als bekannt
Stochastisches math. Modell: Parameter Werte sind mit Wahrscheinlichkeitsverteilungen definiert

Optimierung
Beispiel Müsliherstellung

Rezept:

	Standardmüsli	Früchtemüsli
Haferflocken	0,8 kg	0,8 kg
Trockenfrüchte	0,1 kg	0,2 kg
Nüsse	0,1 kg	

Erlös:
Standardmüsli: 6€/ kg
Früchtemüsli; 8€/ kg

Zutaten:
80 kg Haferflocken
16 kg Trockenfrüchte
7 kg Nüsse

Mathematische Modellierung:

Zielfunktion:

$$\max 6\, x_1 + 8x_2$$

subject to (s.t.) :

$$0{,}8\, x_1 + 0{,}8\, x_2 \leq 80$$
$$0{,}1\, x_1 + 0{,}2\, x_2 \leq 16$$
$$0{,}1\, x_1 \leq 7$$

$$x_1 \geq 0, x_2 \geq 0$$

x_1… zu produzierende Standardmüsli (in kg)
x_1… zu produzierende Menge Früchtemüsli (in kg)

Lineare Optimierung Allgemein

$$\max_x p'x$$

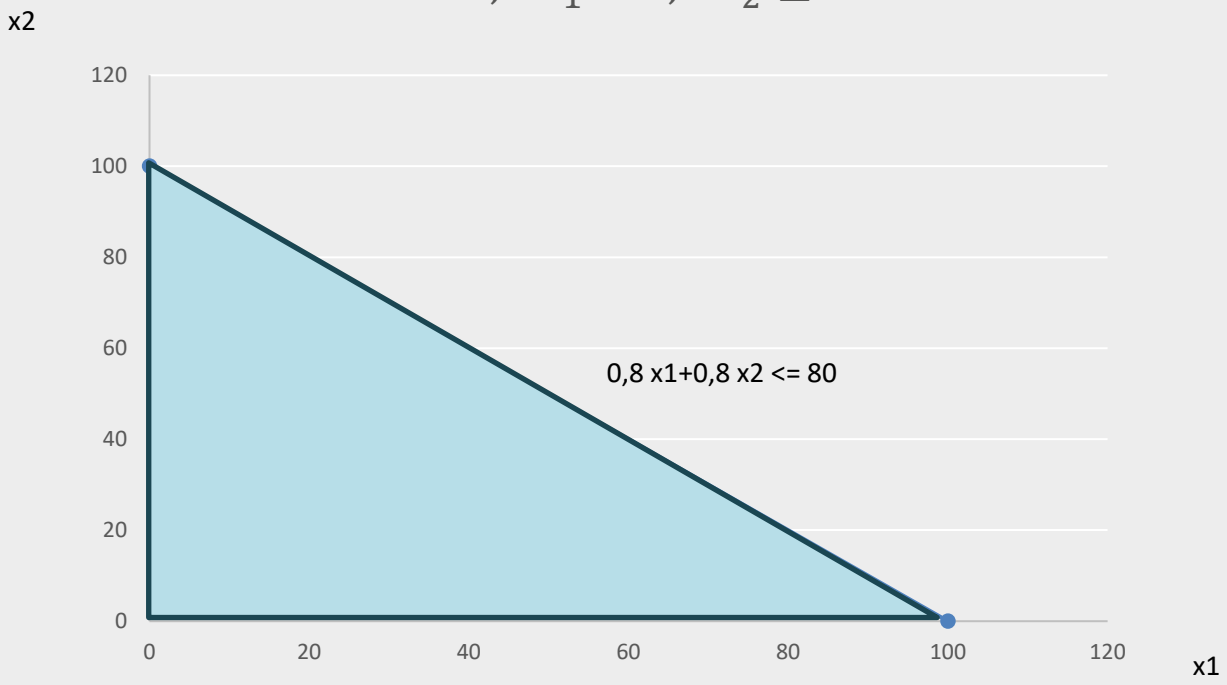
$$Ax \leq b$$
$$x \geq 0$$

p … Erlösvektor
a_{ij} … Menge der Ressource j, die für die Erzeugung von Gut x_i benötigt wird
b_j … verfügbare Menge von Ressource j

Grafische Lösung: zulässiger Bereich

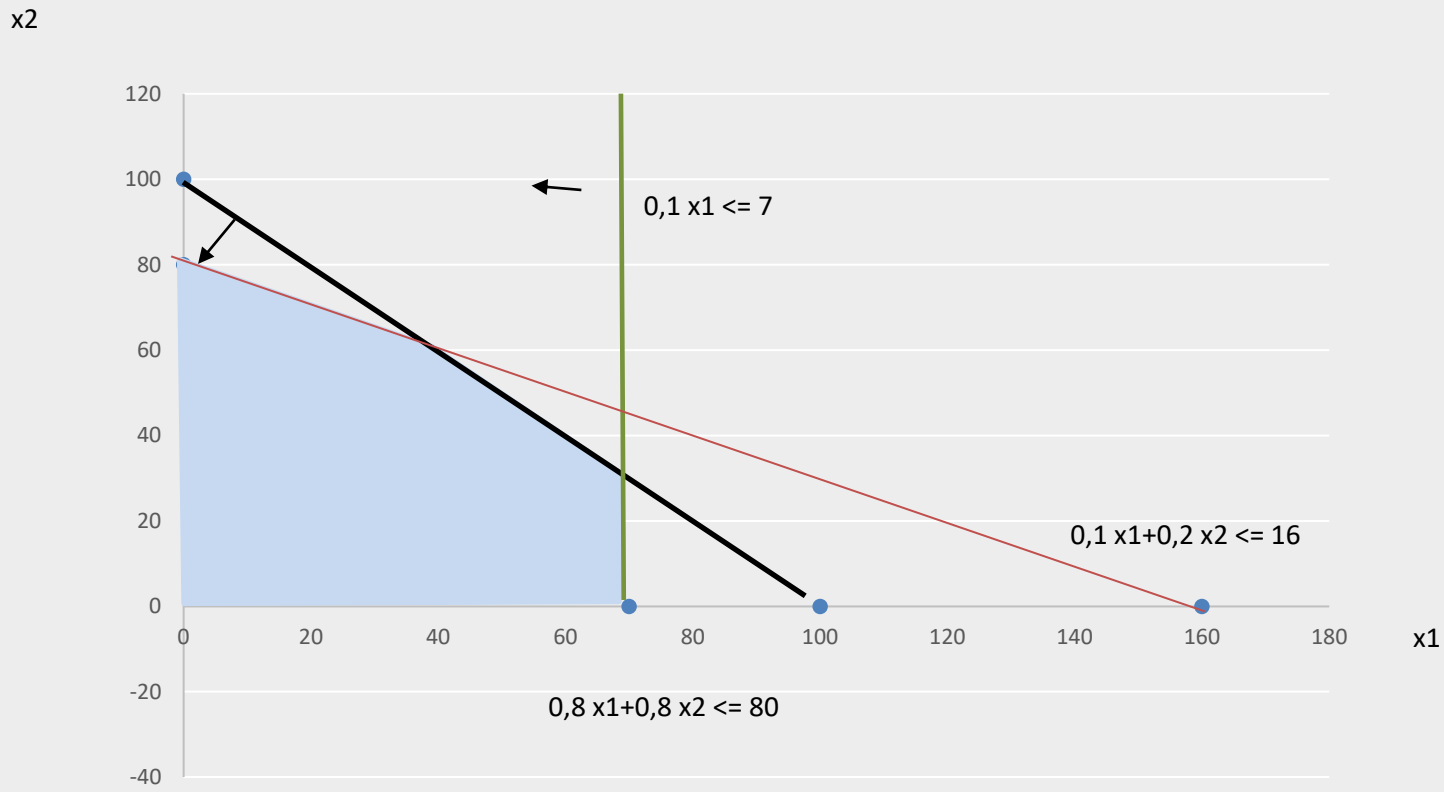
Nebenbedingungen:

$$x_1 \geq 0, x_2 \geq 0$$
$$0{,}8\, x_1 + 0{,}8\, x_2 \leq 80$$

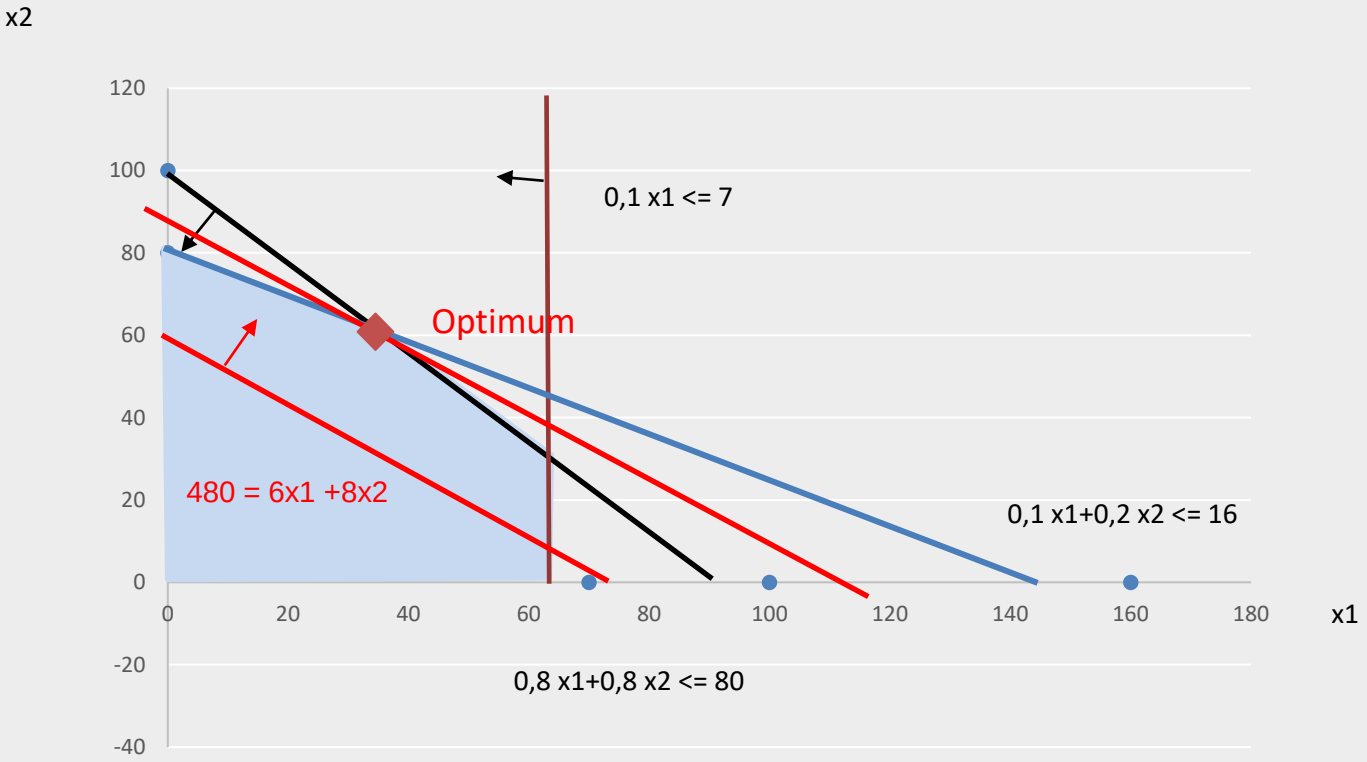


Zulässiger Bereich = Menge aller Werte für die die Entscheidungsvariablen zulässig sind und alle Nebenbedingungen erfüllen

Grafische Lösung: zulässiger Bereich



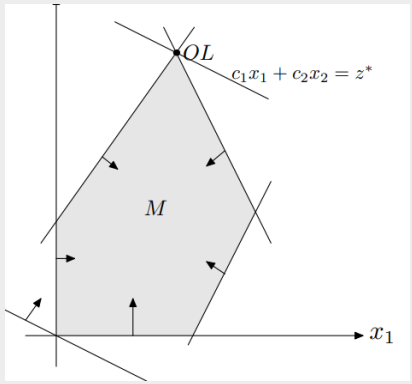
Grafische Lösung: Optimale Lösungsfindung



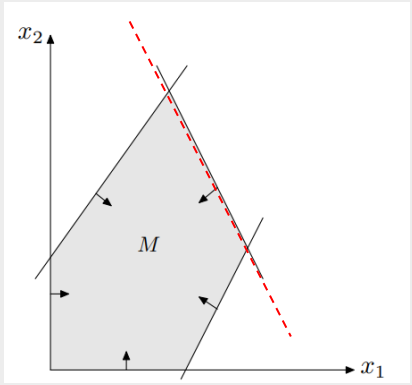
Optimale Lösung = zulässige Lösung ,die mindestens gleich gut ist wie alle anderen zulässigen Lösungen

Gradient der Zielfunktion: Zeigt die stärkste Aufstiegsrichtung an

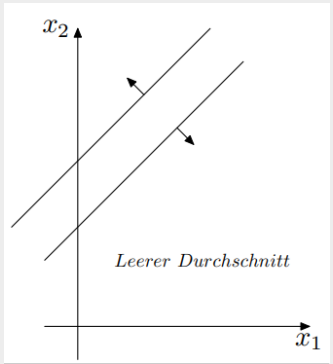
Besondere Lösungen



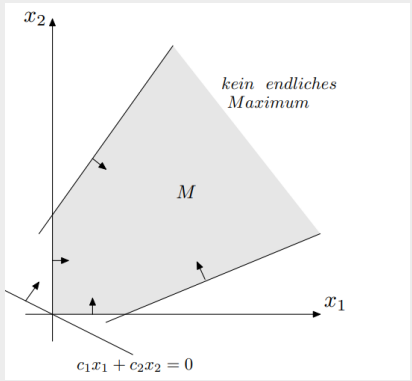
Eindeutige Lösung



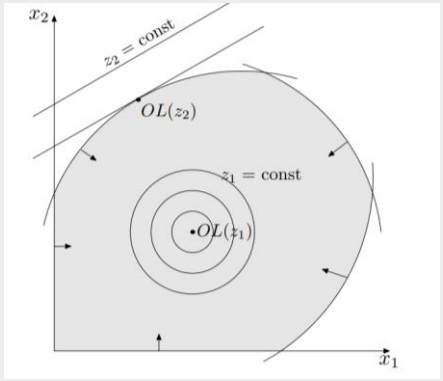
Lösungsmenge



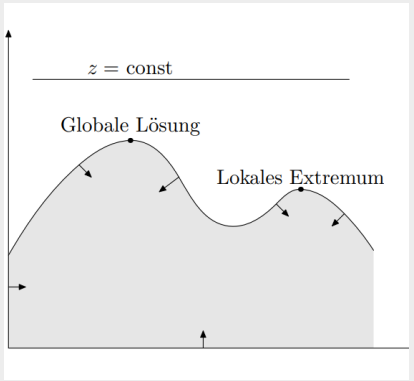
Eindeutige Lösung



Kein endliches Maximum



Konvexe Optimierung



Nicht Konvexe Optimerung mit lokalem Maximum

Programmierung bei zerlegbaren Funktionen

Voraussetzung:
konvexes Optimierungsproblem:

$$\begin{array}{lll} \max & f(x) & konkav \\ s.t. & g_i(x) & konvex \end{array}$$

Wobei f und g_i zerlegbar sein müssen:
$f(x) = \sum_{j=1}^{n} f_j(x_j), \quad g_i(x) = \sum_{j=1}^{n} g_{ij}(x_j)$

Zerlegbarkeit von f(x) impliziert Additivität. Das heißt, es gibt keine Wechselwirkung (Kreuzproduktterme) zwischen den verschiedenen Aktivitäten.
Die Konkavität der fj(xj) bedeutet, dass der Grenzgewinn (die Steigung der Gewinnkurve) niemals steigt, wenn xj zunimmt.

Umformung in ein LP

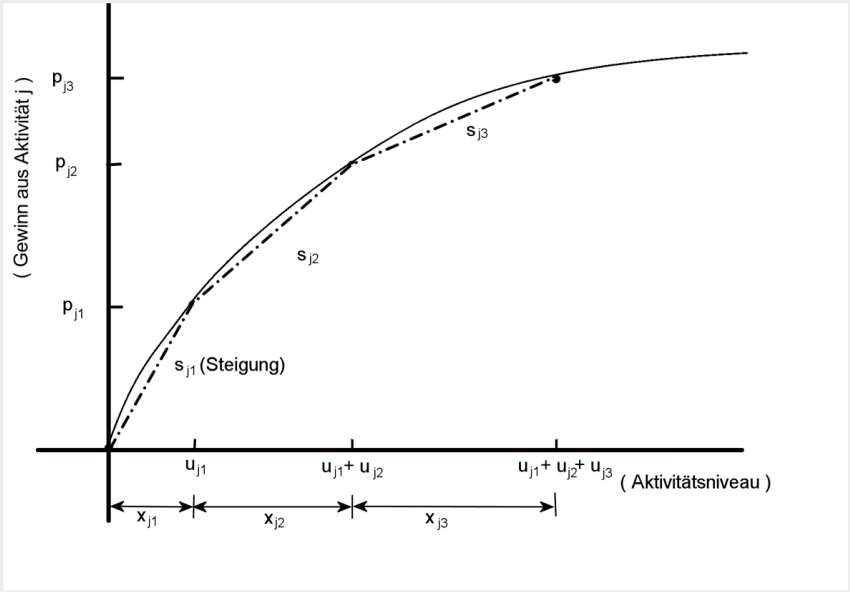
Ang.: nur f(x) wäre nicht linear

Idee: Umwandlung in stückweise lineare Funktion

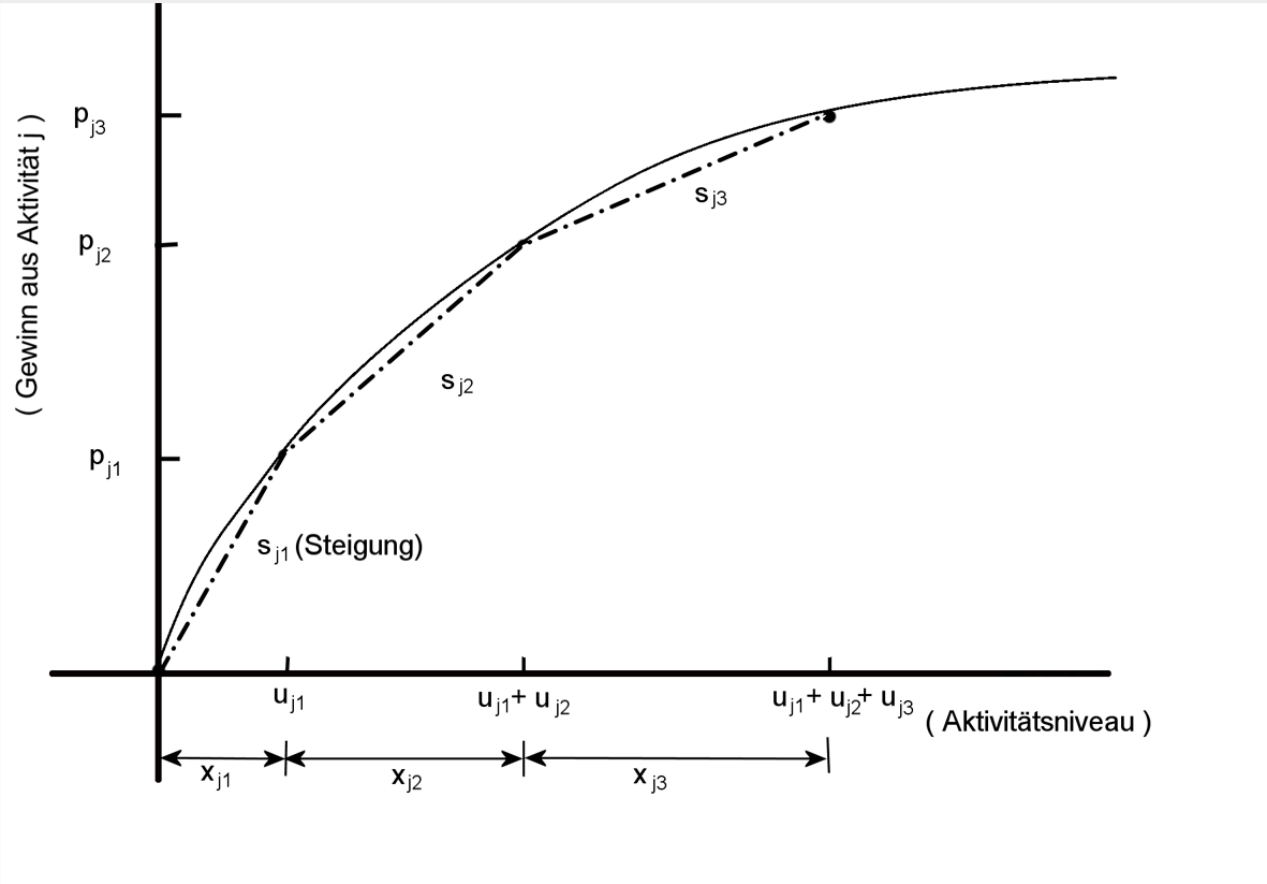
$$x_j = \sum_k x_{jk}$$

$$f_j(x_j) \cong \sum_k s_{jk} x_{jk}$$

$$x_{jk} = 0, \qquad wenn \qquad x_{ji} < u_{ji} \;\; \forall k > i$$



Grafik – Zerlegbare Zielfunktion



→ $s_{j1} > s_{j2} > \ \ldots \ > s_{jk}$

→ Algorithmus füllt automatisch die x_{ji} richtig auf

Schlüsseleigenschaft der Parziell Zerlegbaren Funktionen

Erfüllen $f(x)$ und $g_i(x)$ die Voraussetzungen der PZF, und werden die jeweiligen stückweise linearen Funktionen zu linearen Funktionen umgeformt, dann ergibt sich unter Vernachlässigung der Spezialrestriktion ein Modell der LP, dessen optimale Lösung automatisch die Spezialrestriktion erfüllt.

Erweiterung: Nicht konvexe Optimierungsprobleme

Liegt kein konvexes Optimierungsproblem vor, so kann man theoretisch auch die Funktionen linear approximieren, allerdings muss die Spezial - Nebenbedingung explizit erfüllt sein.

$$x_{jk} = 0, \qquad wenn \qquad x_{ji} < u_{ji} \;\; \forall k > i$$

In Optimierungsmodellen wird diese Nebenbedingung erfüllt, wenn wir hierfür die Variablen des Typs SOS2 (Special Orderd Sets of type 2) einführen.

SOS2- Variablen

SOS Typ 1 -Variablen

Ein Special Ordered Set vom Typ 1 beinhaltet 01-Variablen oder kontinuierliche Variablen mit einem Wertebereich zwischen Null und Eins von denen maximal eine Variable ungleich Null sein darf:

$$\sum_{j=1}^{n} x_j \leq 1$$

SOS Typ 2 - Variablen

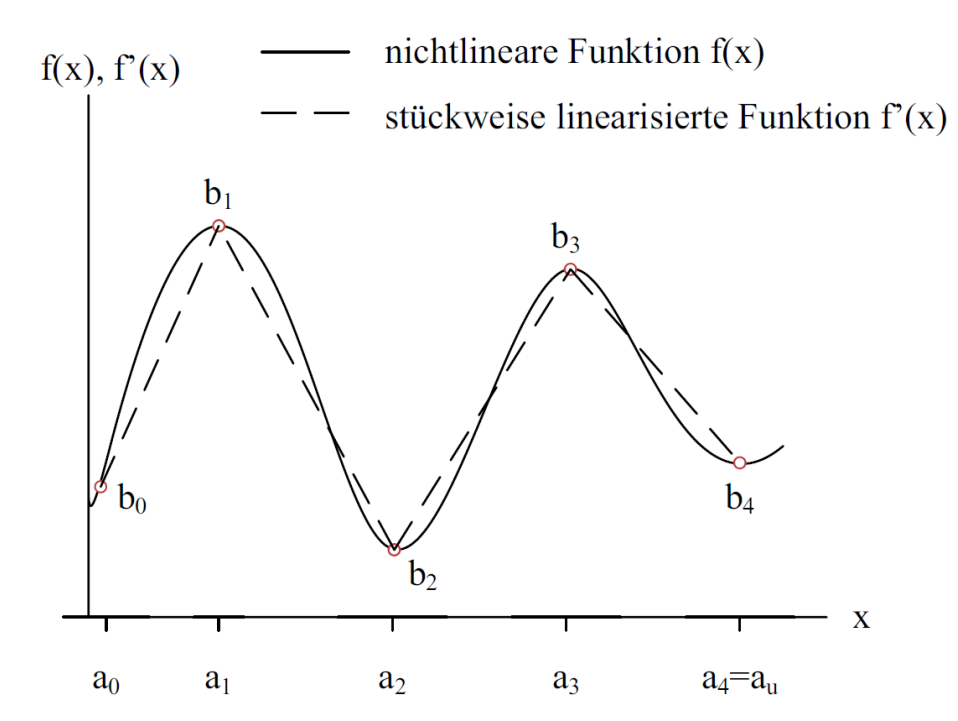
Ein Special Ordered Set vom Typ 2 beinhaltet kontinuierliche Variablen mit einem Wertebereich zwischen Null und Eins von denen maximal zwei Variablen ungleich Null sein dürfen. Sind zwei Variablen ungleich Null, dann müssen diese benachbart sein:

$$\sum_{j=1}^{n} x_j \leq 1$$

Modellierung von nichtlinearen separablen Funktionen

$$f(x_1, \dots, x_n) = \sum_{j=1}^{n} f_j(x_j) \quad \dots separable\ Funktion$$

Modellierung von $f_j(x_j)$ nicht linear:



Funktion f_j wird in 4 lineare Funktionen geteilt:

im Segment k bis k+1 gilt: $x \in [a_k, a_{k+1}]$ mit $y \in [0,1]$:

$$x = a_{k+1}y + a_k(1-y) \rightarrow x - a_k = (a_{k+1} - a_k)y \qquad (1.1)$$

$$f^*(x) = \mathrm{b_k} + \frac{b_{k+1} - b_k}{a_{k+1} - a_k}(x - a_k)$$

$\rightarrow f^*(x) = \mathrm{b_k} + (b_{k+1} - b_k)y = b_{k+1}\,y\, - b_k(1-y)$

Alternativ:

$$x = a_{k+1}z_{k+1} + a_k z_k$$
$$f^*(x) = b_{k+1}\,z_{k+1} - b_k z_k$$

$$z_{k+1} + z_k = 1$$
$$z_k\,, z_{k+1} \geq 0$$

Allgemein:

$$f^*(x) = \sum_{k=0}^{n} b_k z_k$$

$$x = \sum_{k=0}^{n} a_k z_k$$

$$\sum_{k=0}^{n} z_k = 1$$

wobei nur zwei benachbarte z_k ungleich 0 sein dürfen

Deterministische Dynamische Programmierung

Dynamische Programmierung (DP) = math. Methode zur Lösung sequentieller (mehrstufiger) Entscheidungsprozesse.

Steuerung der Strategie zu diskreten Zeitpunkten oder laufend (kontinuierlich)

Formulierung oft schwierig – existiert keine Standardform

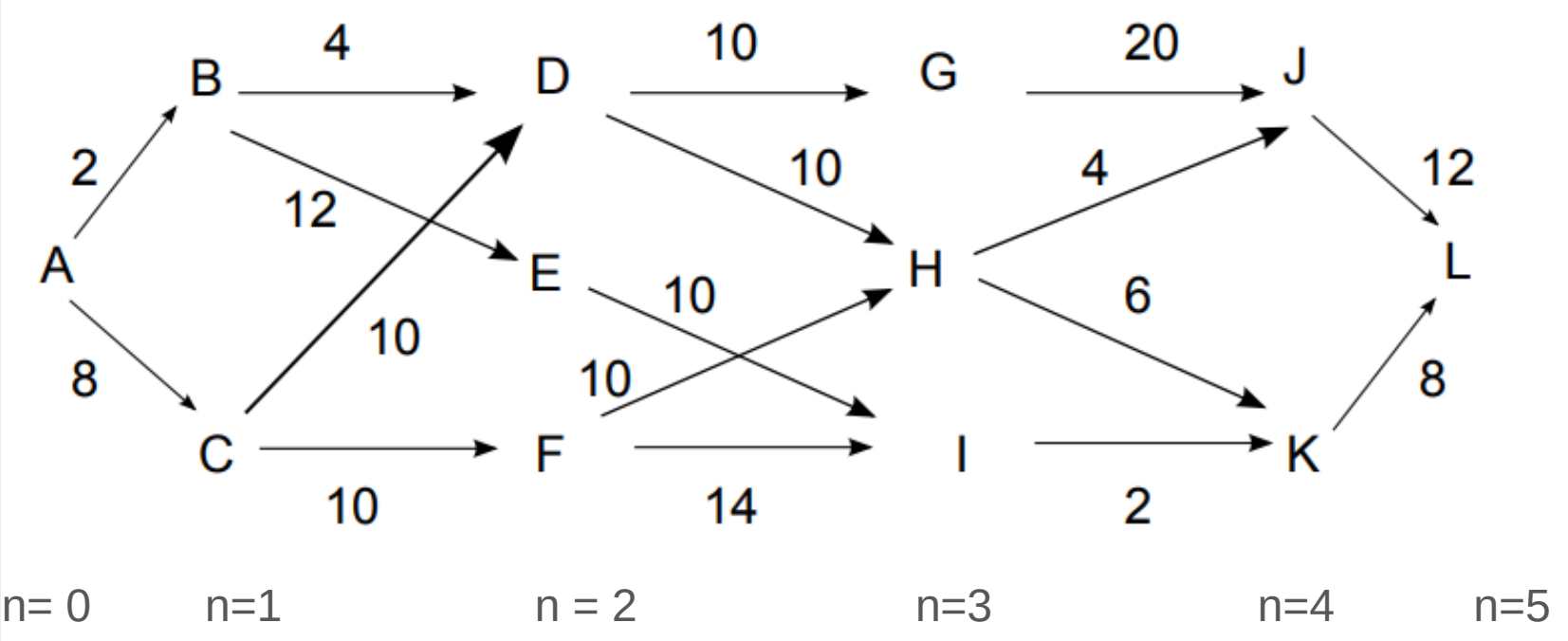
Bsp: Lagerhaltungsprobleme, Produktionsplanung, Planung chemischer Prozesse…

Gründung: Bellman in den 50er Jahren

Dynamische Programmierung
Bsp: Problem des kürzesten Weges

Ein Paket muss von A nach L geliefert werden , Gesucht: kürzester Weg

Lösungsansätze:
1. Enumeration
2. Dynamische Programmierung



Lösungsansatz: Enumeration:

Für jeden Weg wird die Gesamtlänge des Weges bestimmt und dann wir aus diesen Werten der kürzeste ausgewählt.

Weg	Länge
A-B-D-G-J-L	2+4+10+20+12= 48
A-B-E-I- K-L	2+12+10+2+8=34
A-B-D-H-K-L	2+4+10+6+8=30
A-C-F-I-K-L	8+10+14+2+8=42
A-C-D-G-J-L	8+10+10+20+12=60
A-C-F-H-J-L	8+10+10+4+12=44

Nachteil: Kann sehr viele Berechnungen benötigen

Lösungsverfahren: Dynamische Optimierung

Nütze Stufenstruktur– Beginn am Ende der Lösung

Graph hat 5 Stufen (n=0,..,5)
Entscheidungen auf 4 Stufen (n=0,..,4)

Zerlegung des großen Problems in viele kleine:

- Angenommen wir sind auf Stufe 4: Triviale Entscheidung je nachdem in welchem Knoten man ist
  v4(J) = 12
  v4(K) = 8

- Angenommen wir sind auf Stufe 3:
  v3(G) = 20 + v4(J) = 32
  v3(H) = min {4 + v4(J), 6 + v4(K)} = 14 ... Ort K
  v3(I) = 2 + v4(K) = 1

- Stufe 2:
  v2(D) = min {10 + v3(G), 10 + v3(H)} = 24 .. Ort H
  v2(E) = 10 + v3(I) = 20 .. Knoten: I
  v2(F) = min {10 + v3(H), 14 + v3(I)} = 24 ... Ort I

- Stufe 1:
  v1(B) = min {4 + v2(D), 12 + v2(E)} = 28 … Ort D
  v1(C) = min {10 + v2(D), 10 + v2(F)} = 34 .. Ort D, F

- Stufe 0:
  v0(A) = min {2 + v1(B), 8 + v1(C)} = 30 … Ort B

Lösung. A-B-D-H-K-L

Prinzip der Dynamsichen Optimierung

Zerlegung eines mehrstufigen Problems in viele einstufige Teilprobleme

Das Verfahren der Dynamischen Optimierung hat 2 Phasen:

1. Rückwärtsrekursion:
   Ermittlung auf jeder Stufe in jedem Knoten (bzw.Zustand) den kürzesten Weg und Notation der optimale Bewertung für den Restweg
2. Vorwärtsrechnung:
   Beginnend beim Ausgangspunkt wird optimlale Route zusammen gesetzt.

Optimalitätsprinzip von Bellman:
Sei $(x_0^*, x_1^*, \dots, x_j^*, \dots, x_{n-1}^*)$ eine optimale Lösung, die das System vom Anfangszustand $z_0 = a$ in den Endzustand $z_n = b$ überführt, wobei das System zum Zeitpunkt j den Zustand z_j^* annimmt. Dann gilt: $(x_j^*, \dots, x_{n-1}^*)$ ist eine optimale (Teil-)Lösung, die das System vom vorgegebenen Zustand z_j^* in den Endzustand b überführt.

$$F = \sum_{k=0}^{n-1} r_k(z_k, x_k) + r_n(z_n)$$

oder mit anderen Worten:
eine optimale Lösung hat die Eigenschaft, dass unabhängig vom Anfangszustand und den anfänglichen Entscheidungen die verbleibenden Entscheidungen ausgehend vom aktuellen Zustand optimal sind.

GAMS

General Algebraic Modeling System

Download:

https://www.gams.com/download/

GAMS Modellierung

Verallgemeinerung unseres Modells:

$$\max 6\,x_1 + 8x_2$$

subject to (s.t.) :

$$0{,}8\,x_1 + 0{,}8\,x_2 \leq 80$$
$$0{,}1\,x_1 + 0{,}2\,x_2 \leq 16$$
$$0{,}1\,x_1 \leq 7$$

$$x_1 \geq 0, x_2 \geq 0$$

Allgemein:

$$Z = \max \sum_j c_j\,x_j$$

$$s.t. \quad \sum_j a_{ij}\,x_j \leq b_i \qquad \forall\, i \in I$$

$$x_j \geq 0 \qquad \forall\, j \in J$$

Mengen
I, J

Parameter
c_j, a_{ij}, b_i

Variablen
Z … Zielfunktionsvariable
x_j… Variablen

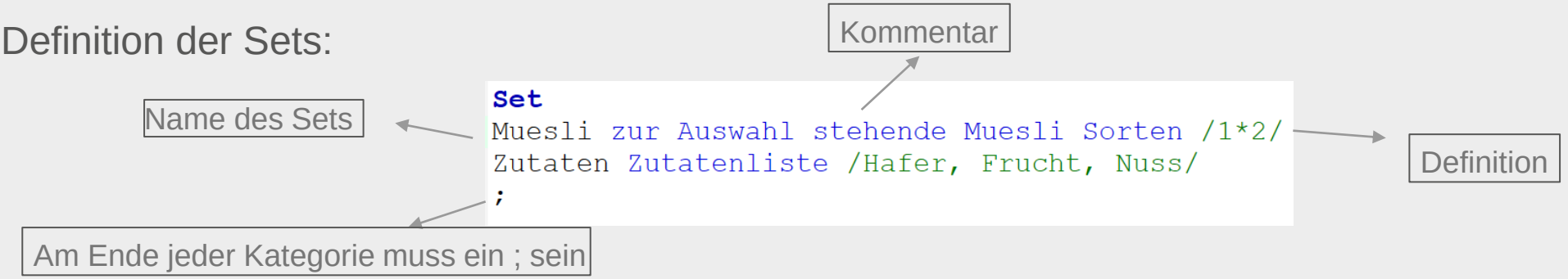
Gleichungen

GAMS Syntax

Math. Modell	GAMS Syntax	
Mengen	sets	(Daten)
Parameter	parameters tables scalars	(Daten)
Variablen	variables	(Modellstruktur)
Zielfunktion	equations	(Modellstruktur)
Nebenbedingungen	equations	(Modellstruktur)

GAMS Code

Definition der Sets:



Definition der Parameter:

```
Parameter
DB(Muesli)/
1       6
2       8
/

Zusammensetzung(Muesli, Zutaten) Matrix der Zutaten je Muesli/
1.Hafer=0.8
1.Frucht=0.1
1.Nuss=0.1
2.Hafer=0.8
2.Frucht=0.2
2.Nuss=0
/


Lager(Zutaten) Lagerbestand aller Zutaten/
Hafer    80
Frucht    16
Nuss     7
/

;
```

Tipp:
Verwendung von sprechenden Namen für Sets, Parameter Variablen und Gleichungen

Achtung:
GAMS ist nicht Case-sensitiv

GAMS Code

Definition der Variablen

```
Variable
v_Erloes
;

Positive Variable

v_Produktion(Muesli)    optimale Produktionsmenge je Muesli
;
```

Tipp:
Bei Benennung von Variablen mit v_XY ist in den Gleichungen gleich gut ersichtlich ob man im Linearen Optimierungsproblem bleibt

Definition der Zielfunkton

```
Equation
ZF  Zielfunkton

Rezept   Zusammensetztung je Müsli und einhaltung des Lagerbestandes

;

ZF..          v_Erloes =e= sum(Muesli, v_Produktion(Muesli)*DB(Muesli) );
Rezept(Zutaten)..          sum(Muesli, Zusammensetzung(Muesli, Zutaten)*v_Produktion(Muesli) )=l= Lager(Zutaten);
```

Operatoren

GAMS	Operator
=e=	=
=l=	≤
=g=	≥

GAMS Code

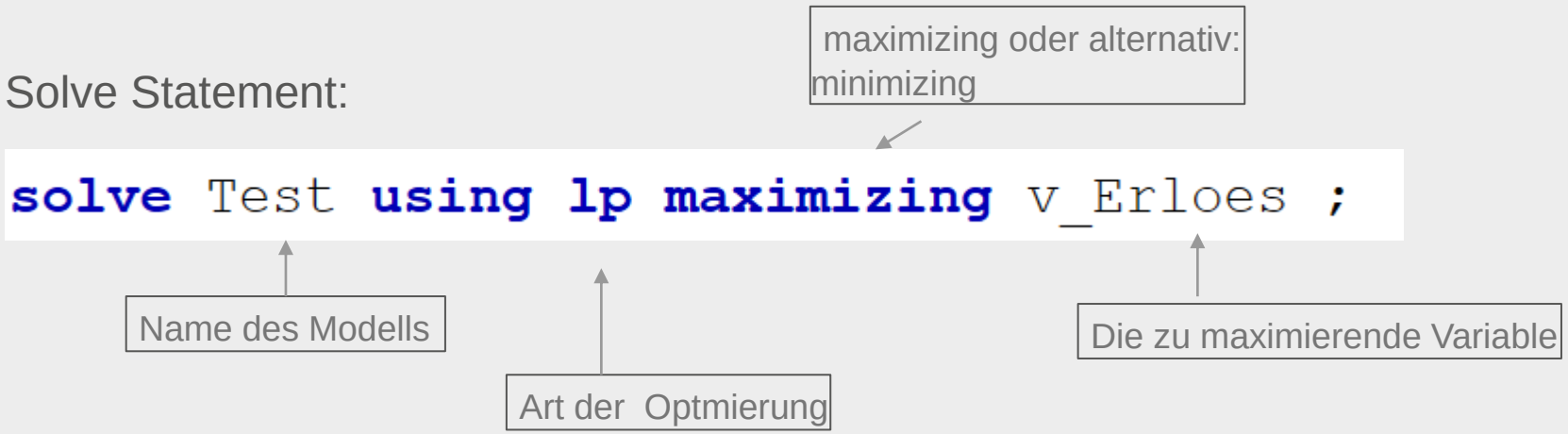
Beschreibung des Optimierungsmodells:

```
Model Beispiel1 /all/;
```

Alternativ (wenn nicht alle Gleichungen/ NB ins Modell sollen):

```
Model Test /ZF, Rezept/;
```

Solve Statement:



Modelltypen und Solver

GAMS Syntax	Modelltyp
lp	Linear program
nlp	Nonlinear Program
Qcp	Quadratically Constrained Program
mip	Mixed Interger Program
rmip	Relaxed mixed Integer Program
minlp	Mixed Integer Nonlinear Program
…	

Bem.: Die LP-Relaxation (relaxed mixed integer program) eines ganzzahligen Programs wird als rmip bezeichnet. Dabei wird die Ganzzahligkeitseigenschaft der Variablen verworfen.

Erweiterete GAMS Syntax
Sets

Einem Set dürfen mehrere Namen gegeben werden (alias)

```
set i Fabriken /1*5/
;
alias (i,k,l)
```

Subsets:

```
p(i) Papierfabrik /1*2/
```

Mehrdimensionale Sets:

```
w(i,i) Wege zwischen den Fabriken /
1.2
1.3
2.4
3.4
4.5
/
```

Erweiterete GAMS Syntax

Funktionen:

t… Parameter
card(t) …. Kardinalität
ord(t)… Ordnung

Verwendung: gewisse Gleichung soll nur für den letzten Parameter nicht gelten:

```
g_Lagergleichung(t)$(ord(t)<card(t))...  v_Lager(t+1)=e=v_Lager(t) +v_Einkauf(t+1)
```

Bem. : Will man die Gleichung Zyklisch machen : Verwendet man t++1 statt t+1 , so ist der Nachfolger des letzten Elements das erste Element

$-Bedingungen:

Logische Operatoren	
Operator	Bedeutung
not	nicht
and	und
or	oder
xor	Exjkusiv oder

Numerische Operatoren		
Operator		Bedeutung
lt	<	Kleiner als
le	<=	Kleiner oder gleich
eq	=	Gleich
ne	<>	Ungleich
ge	>=	Größer oder gleich
gt	>	Größer

V

Weiter Funktionen:

Summe und Produkt

```
*Beispiel zur Summenbildung: sum()
 sum(j, x(j) - y(j) );
* Eine Summe über 2 Mengen -
* Indices müssen eingeklammert werden:
 sum((k,l), z(k,l) * p(k) );
*Beispiel zur Produktbildung:
 prod(j, p(j) * r(j) );
```

Parameter

Man kann diverse Rechnungen im GAMS ausführen

z.B Maximum diverser Werte:

```
Parameter
m1 Maximum
;
m1= max(1,10, 55*2);
```

Oder Maximum eines Parameters über ein Set i

```
set
i Fabriken /1*5/;
Parameter
c(i)  Testdaten  /1,3,10,-2/
m1
m2
;
m1=smax(i,c(i));
m1=smin(i,c(i));
```

Variablen

GAMS	Bedeutung
.lo	untere Schranke (lower bound)
.up	obere Schranke (upper bound)
.fx	Fixierung: untere und obere Schranke sind identisch

Vorteil: kann für den Solver vorteilhaft sein
Bsp.: x.up(set) =1

Fehlerhinweise - Logfile

Logfile (.log-Datei)

- Fehlermeldungen werden mit Zeilenangabe in der Reihenfolge ihres Auftretens im logfile-Fenster angezeigt
- Doppelklick auf die rot markierte Fehlermeldung führt zur fehlerhaften Zeile im .gms File
- Ein Fehler führt oft zu mehreren Fehlermeldungen → immer zuerst die erste Fehlermeldung bearbeiten!