

Methoden und Anwendungen der Optimierung

WS 2015 / 2016

Prof. Dr. Hans-Jürgen Sebastian

Vorlesungstermine WS 2015 / 2016

Datum	Wochentag	Zeit	Hörsaal
20.10.	Dienstag	14:15-15:45 h	III
21.10.	Mittwoch	08:30-10:00 h	III
03.11.	Dienstag	14:15-15:45 h	III
04.11.	Mittwoch	08:30-10:00 h	III
17.11.	Dienstag	14:15-15:45 h	III
18.11.	Mittwoch	08:30-10:00 h	III
01.12.	Dienstag	14:15-15:45 h	III
02.12.	Mittwoch	08:30-10:00 h	III
15.12.	Dienstag	14:15-15:45 h	III
16.12.	Mittwoch	08:30-10:00 h	III
19.01.	Dienstag	14:15-15:45 h	III
20.01.	Mittwoch	08:30-10:00 h	III
26.01.	Dienstag	14:15-15:45 h	III
27.01.	Mittwoch	08:30-10:00 h	III

Klausuren

PT1: 19.02.2016

PT2: 16.03.2016

1. Diskrete und kombinatorische Optimierung
 - 1.1. Standardprobleme der kombinatorischen Optimierung und Relaxationen, Rucksack- und Bin-Packing-Probleme, Set-Covering-, Set-Packing- und Set-Partitioning Probleme, Fixkosten- oder Fixed-Charge-Probleme, Relaxationen
 - 1.2. Gemischt-ganzzahlige Optimierung und die Methode Branch and Bound
 - 1.3. Branch and Bound für binäre Optimierungsprobleme
 - 1.4. Allgemeine Darstellung der Methode Branch and Bound
 - 1.5. Schnittebenen-Verfahren: Das Verfahren von Gomory
 - 1.6. Ausblick: Branch-and-Cut-Verfahren
2. Heuristiken und Metaheuristiken
 - 2.1. Greedy Algorithmen
 - 2.2. Lokale Suche
 - 2.3. Simulated Annealing
 - 2.4. Tabu Search
 - 2.5. Genetische und Evolutionäre Algorithmen

Inhaltsübersicht

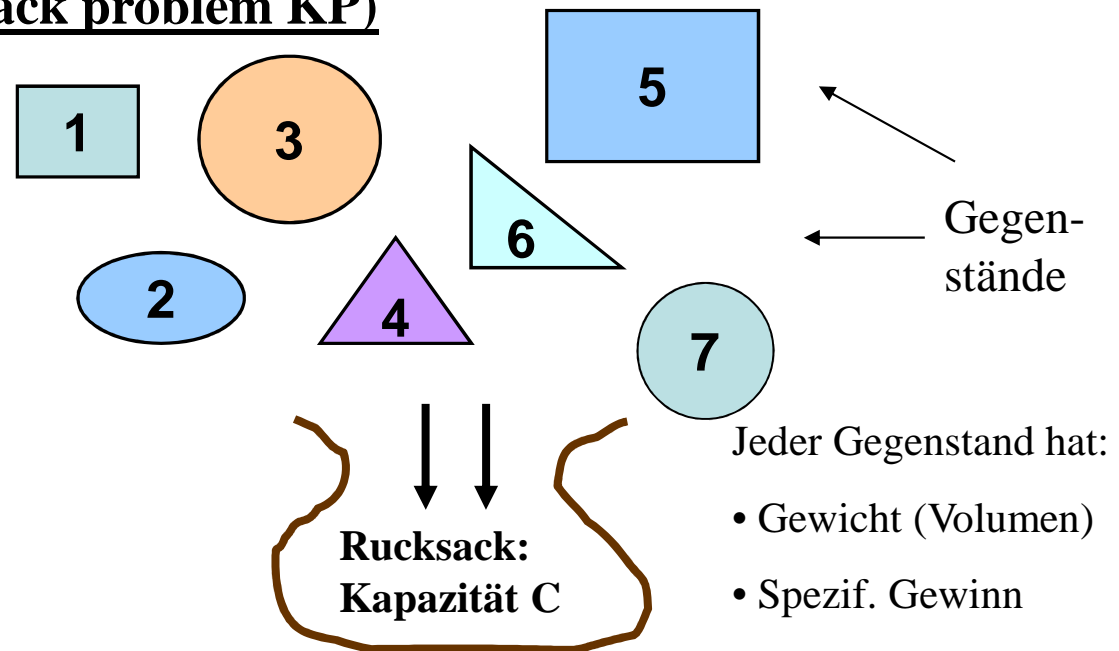
- 3. Flüsse in Netzwerken, Transport- und Tourenplanung
 - 3.1. Netzflußprobleme: Minimalkosten-Netzflußproblem, Kürzeste-Wege-Problem, Mehrgüterflüsse
 - 3.2. Transportproblem
 - 3.3. Das Vehicle Routing Problem(VRP): Modellierung und heuristische Lösung
 - 4. Nichtlineare Optimierung
 - 4.1. Grundlegende Begriffe und Eigenschaften
 - 4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen
 - 4.3. Lagrangefunktion und Optimalitätsbedingungen
 - 4.4. Quadratische Probleme und das Verfahren von Wolfe
 - 4.5. Dualität in der Nichtlinearen Optimierung
 - 4.6. Ausgewählte Numerische Verfahren
 - 4.7. Lagrange Relaxation und Subgradientenoptimierung
 - 5. Dynamische Optimierung und Lagerhaltung
 - 5.1. Mehrstufige Entscheidungsprozesse
 - 5.2. Bellmann'sches Optimalitätsprinzip und Funktionalgleichungen
 - 5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)
-

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen:

Rucksack- und Bin-Packing-Probleme, Set-Covering-, Set-Packing- und Set-Partitioning Probleme, Fixkosten- oder Fixed-Charge-Probleme, LP-Relaxation

Rucksackprobleme (Knapsack problem KP)

Welche Gegenstände in einen Rucksack begrenzter Kapazität packen, damit „Gewinn“ möglichst groß wird.



1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Abstraktion: Eine Ressource begrenzter Kapazität soll möglichst effizient genutzt werden.

Modellierung: Binäres Optimierungsproblem

Gegenstände j: $j = 1, 2, \dots, n$

Binäre Entscheidungsvariable x_j :

$$x_j = \begin{cases} 1, & \text{falls Gegenstand mit-} \\ & \text{genommen wird} \\ 0, & \text{sonst} \end{cases}$$

Daten: Gewinn p_j , Gewicht w_j von Gegenstand j
 C Kapazität des Rucksacks

$$p_j > 0, w_j > 0, C > 0$$

Modell: Maximiere $Z_{KP} = \sum_{j=1}^n p_j x_j$ (1)

so, dass: $\sum_{j=1}^n w_j x_j \leq C$ (2)

$x_j \in \{0, 1\}$ für alle $j = 1, 2, \dots, n$ (3)

Binäre EV, eine Restriktion, “Lineare” Zielfunktion

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Varianten des KP: Bisher Binäres Rucksackproblem

- beschränktes Rucksackproblem: man kann bis zu b_j (ganze Zahl) Einheiten eines Gegenstandes mitnehmen
→ $x_j \in \{0, 1, \dots, b_j\}$
- kontinuierliche Rucksackprobleme
LP-Relaxation → $0 \leq x_j \leq 1$ für alle $j = 1, 2, \dots, n$

Lösung des kontinuierlichen KP: (Greedy Algorithmus)

1. Sortiere Gegenstände nach fallendem “Gewinn pro Gewichtseinheit“ p_j/w_j
2. Auffüllen des Rucksacks solange bis letzter Gegenstand nicht mehr hineinpasst.
(ZF – Wert des kontinuierlichen KP ist obere Schranke für ZF – Wert Z_{KP} des binären KP.)
3. Der Gegenstand der nicht mehr vollständig in den Rucksack hineinpasst heißt kritischer Gegenstand.

$$s = \min \left\{ j \in \{1, \dots, n\} : \sum_{i=1}^j w_i > C \right\}$$

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

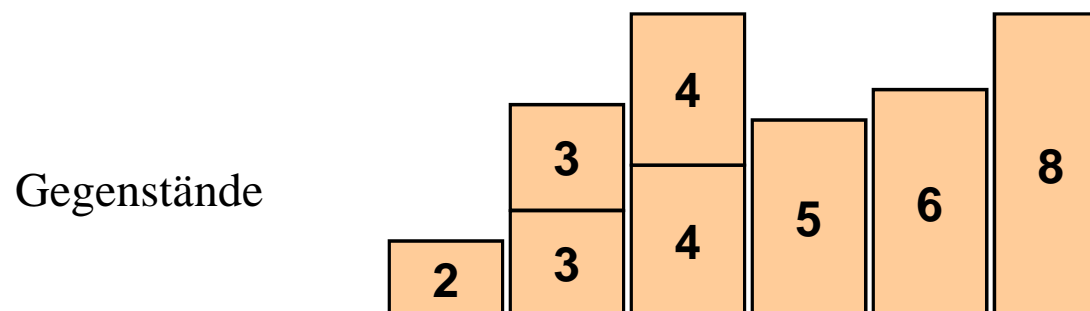
4. Von Gegenstand s anteilig soviel einpacken, bis Rucksack voll ist.

$$\bar{x}_j = \begin{cases} 1 & , \text{für } j = 1, \dots, s-1 \\ (C - \sum_{j=1}^{s-1} w_j) / w_s & , \text{für } j = s \\ 0 & , \text{für } j = s+1, \dots, n \end{cases} \quad (4)$$

Die \bar{x}_j , $j = 1, \dots, n$ stellen die Lösung der [LP-Relaxation des Rucksackproblems](#) dar.

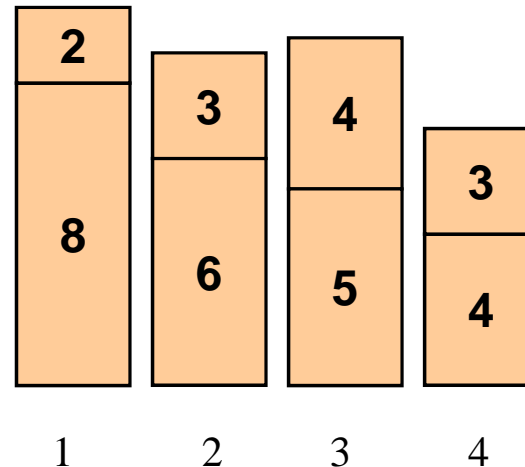
Das Bin-Packing Problem

Beispiel: 8 Gegenstände seien auf Behälter der Kapazität $C = 10$ so aufzuteilen, dass eine minimale Anzahl von Behältern benötigt wird.



1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Zulässige Lösung:



4 Behälter der Kapazität 10 werden benötigt:

Insgesamt benötigte Kapazität der Gegenstände ist 35,
Behälterkapazität ist 10

→ Lösung ist optimal, da 4 Behälter mindestens benötigt werden.

Packprobleme: LKW, Waggon, Flugzeuge, Schiffe möglichst gut auslasten!
Bei Stückgütern Packung der Gegenstände optimieren.

Grundlegendes Problem: [Bin-Packing-Problem](#)

Eine Menge von n Gegenständen soll in eine Menge gleichgroßer Behälter (bins) gepackt werden. Kapazität der Behälter sei $C > 0$.

Jeder Gegenstand i nimmt eine Kapazität von $w_i > 0$, $i = 1, \dots, n$ in Anspruch

Aufgabe: Gegenstände so in Behälter verpacken, dass eine minimale Anzahl von Behältern benötigt wird.

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Modell:

Voraussetzung: Jeder Gegenstand passt in einen Behälter der Kapazität C , d.h. $w_i \leq C$ für alle i . (Dann sind maximal n Behälter erforderlich)

Entscheidungsvariablen:

$$y_j = \begin{cases} 1, & \text{falls Behälter } j \text{ benötigt wird} \\ 0, & \text{sonst} \end{cases}$$
$$j = 1, 2, \dots, n$$
$$x_{ij} = \begin{cases} 1, & \text{falls Gegenstand } i \text{ in Behälter } j \text{ gepackt wird} \\ 0, & \text{sonst} \end{cases}$$

Zielfunktion:

$$\text{Minimiere } Z_{BP} = \sum_{j=1}^n y_j \quad (5)$$

Restriktionen:

so, dass

$$\sum_{i=1}^n w_i x_{ij} \leq C \cdot y_j \quad \text{für alle } j = 1, \dots, n \quad (6)$$

$$\sum_{j=1}^n x_{ij} = 1 \quad \text{für alle } i = 1, \dots, n \quad (7)$$

$$y_j \in \{0,1\} \text{ für alle } j = 1, \dots, n \quad x_{ij} \in \{0,1\} \text{ für alle } i, j = 1, \dots, n \quad (8)$$

Die Zielfunktion minimiert die Anzahl der benötigten Behälter. Restriktion (6) stellt sicher, dass die Zuordnung von Gegenstand i zum Behälter j die Kapazität eines solchen Behälters j nicht übersteigt und dass zu einem nicht benötigten Behälter keine Gegenstände zugeordnet werden (Wenn $y_j=0 \rightarrow$ alle $x_{ij} = 0$, $i= 1, \dots, n$). Restriktion (7) legt zusammen mit (8) fest, dass jeder Gegenstand genau einem Behälter zugeordnet wird.

Es gibt allgemeinere Bin-Packing-Probleme:

- Verschiedene Behältergrößen: $C \rightarrow C_j$
- Mehrdimensionale Charakterisierung der Behälter und Gegenstände (Fläche, Gestalt)

Algorithmische Grundidee der Lösungsverfahren:

- Zunächst “große” Gegenstände verpacken
- kleinere Gegenstände hinzupacken

Set-Covering, Set-Packing und Set-Partitioning

Beispiel

Serviceteams sollen eine Menge von Kunden besuchen. Für diese Teams sollen nun Besuchspläne bei den Kunden entwickelt werden.

Unterschiedliche Prinzipien (Eigenschaften) dieser Besuchspläne:

- | | | |
|--------------------------------------|-------------------------|-----------|
| - Jeder Kunde soll höchstens einmal | Set-Packing | } Problem |
| - Jeder Kunde soll mindestens einmal | Set-Covering | |
| - Jeder Kunde soll genau einmal | Set-Partitioning | |

besucht werden. (Jedes Serviceteam kann jeden Kunden besuchen und es gibt ausreichend viele Serviceteams)

Probleme treten in vielen unterschiedlichen Varianten in der Logistik auf.

Mathematische Formulierung

$M = \{1, \dots, m\}$ sei eine endliche Menge; $M_j \subseteq M, j=1, \dots, n$, seien Teilmengen von M

z.B.: M Menge der zu besuchenden Kunden

M_j Kunden, die zum j -ten Besuchsplan gehören

Auswahl von Teilmengen ist das Ziel. Deshalb kann man eine Lösung durch die Indexmenge $I, I \subseteq \{1, 2, \dots, n\}$ der darin enthaltenen Teilmengen beschreiben.

Zum Beispiel: $n = 10$ Teilmengen M_1, \dots, M_{10} stehen zur Auswahl
 $I = \{1, 4, 5, 9\}$ ist eine Lösung (Auswahl) die die Mengen M_1, M_4, M_5, M_9 enthält.

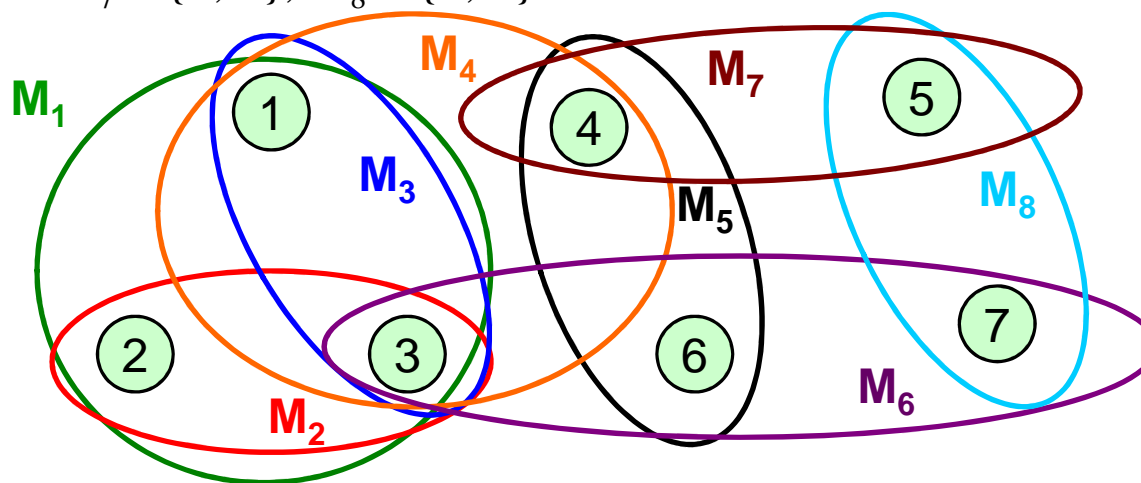
1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Beispiel: $M = \{1, \dots, 7\}$, $m = 7$

$n = 8$ Teilmengen von M : M_1, M_2, \dots, M_8

$M_1 = \{1, 2, 3\}$, $M_2 = \{2, 3\}$, $M_3 = \{1, 3\}$, $M_4 = \{1, 3, 4\}$, $M_5 = \{4, 6\}$, $M_6 = \{3, 6, 7\}$,

$M_7 = \{4, 5\}$, $M_8 = \{5, 7\}$



$I = \{2, 5, 8\}$

ist ein Set Packing

$I = \{1, 6, 7\}$

ist ein Set Covering

$I = \{1, 5, 8\}$

ist ein Set Partitioning

Set – Packing: falls $M_j \cap M_k = \emptyset$ für alle $j, k \in I, j \neq k$

Set – Covering: falls $\bigcup_{j \in I} M_j = M$

Set – Partitioning: falls I sowohl Set – Packing als auch Set – Covering

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Bewertung: Man kann jeder Teilmenge z. B. Kosten oder Gewinn zuordnen.

Danach kann man minimieren bzw. maximieren.

→ **Formulierung als binäres Optimierungsproblem:**

- **Charakteristischer Vektor:** b_j der Menge $M_j \subseteq M$, ist ein m -dimensionaler Vektor der an der Stelle i eine 1 besitzt, falls $i \in M_j$.
Ansonsten ist der Eintrag = 0.
- **Inzidenzmatrix B:** $m \times n$ Matrix, die als j -te Spalte den charakteristischen Vektor der Menge M_j enthält.

Beispiel:

Elemente	Teil- mengen	M1	M2	M3	M4	M5	M6	M7	M8
1		1	0	1	1	0	0	0	0
2		1	1	0	0	0	0	0	0
3		1	1	1	1	0	1	0	0
4		0	0	0	1	1	0	1	0
5		0	0	0	0	0	0	1	1
6		0	0	0	0	1	1	0	0
7		0	0	0	0	0	1	0	1

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Binäre Optimierungsprobleme

$$x_j = \begin{cases} 1, & \text{falls Teilmenge } M_j \text{ zur Lösung gehört} \\ 0, & \text{sonst} \end{cases}$$

$c^T = (c_1, \dots, c_n), \quad x^T = (x_1, \dots, x_n)$ Vektoren der EV bzw. ZF- Koeffizienten

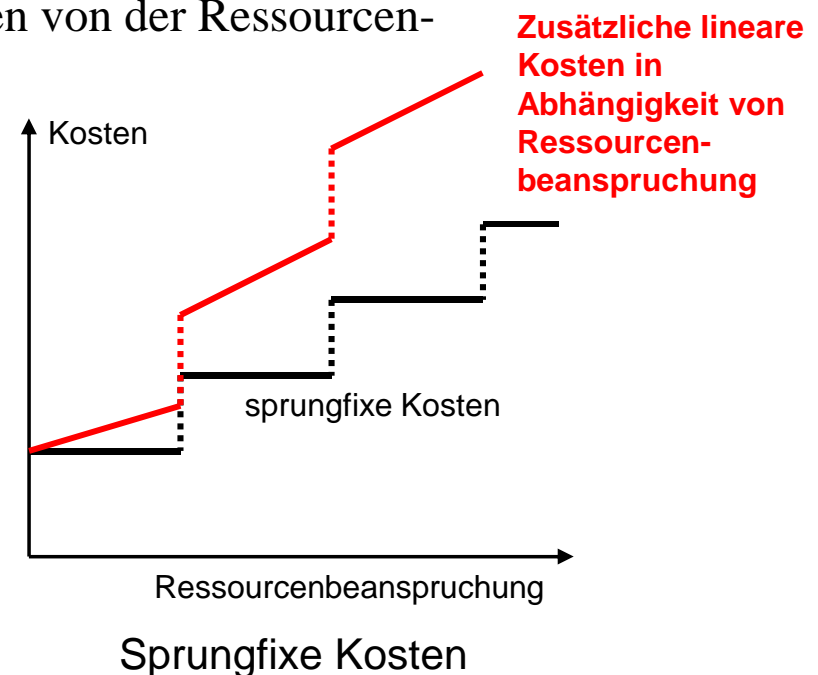
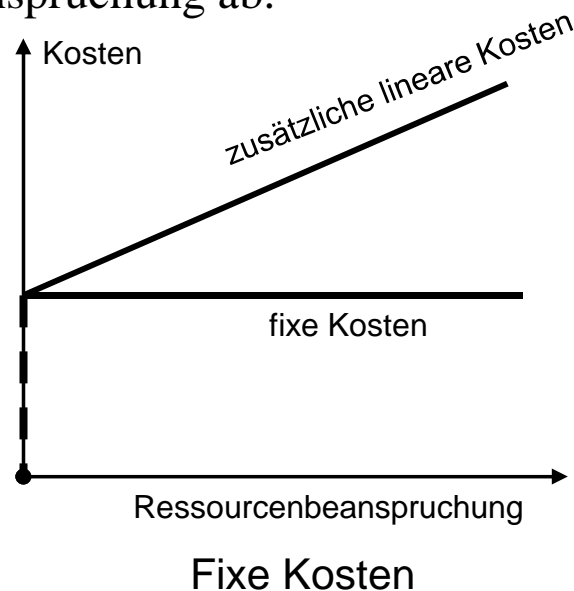
SPK Set - Packing	SC Set - Covering	SP Set - Partitioning
$\underline{\max} Z_{SPK} = c^T x$ so dass $Bx \leq 1$ $x \in \{0,1\}^n$ maximiert Gewinn	$\underline{\min} Z_{SC} = c^T x$ so dass $Bx \geq 1$ $x \in \{0,1\}^n$ minimiert Kosten	$\underline{\min} Z_{SP} = c^T x$ so dass $Bx = 1$ $x \in \{0,1\}^n$ minimiert Kosten
jeweils einer Auswahl		

Heuristische und exakte Verfahren stehen zur Verfügung, die Probleme mit mehreren tausend Teilmengen (Spalten von B) lösen können.

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Das Fixkosten- oder Fixed-Charge-Problem

- Bereitstellung einer Ressource verursacht häufig fixe Kosten (Kauf eines Lagers, Bereitstellung eines Fahrzeugs)
 - fixe Kosten unabhängig von der tatsächlichen Nutzung bzw.
 - durch Aufwendung fixer Kosten bestimmte Kapazitätserweiterungen erreichen (sprungfixe Kosten). Sprungfixe Kosten hängen von der Ressourcenbeanspruchung ab.



1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Beispiel für ein Fixkostenmodell: Facility (Warehouse) Location Problem

Menge von Kunden $M = \{1, 2, \dots, m\}$

Menge von Standorten $N = \{1, 2, \dots, n\}$ (potentielle Standorte)

- Die Kunden sind von den Standorten zu beliefern
- Die Einrichtung von “facilities“ (Produktionsanlagen, Lager, Sortiereinrichtungen etc.) am Standort $j \in N$ verursacht Fixkosten $f_j \geq 0$
- Die Transportkosten für die Belieferung des Gesamtbedarfs des Kunden $i \in M$ vom Standort $j \in N$ seien $c_{ij} \geq 0$ (Zuordnungs- (allocation) Kosten)

Problem: Gesucht ist die kostenminimale Auswahl von Standorten zur Errichtung der “Facilities“ und simultan die kostenminimale Belieferung der Kunden so dass deren Nachfrage komplett befriedigt wird.

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Mathematisches Modell (FLP bzw. WLP)

- Entscheidungsvariablen:

$$y_j = \begin{cases} 1, & \text{falls Standort } j \text{ ausgewählt wird} \\ 0 & \text{sonst} \end{cases}$$

$j \in N$ x_{ij} – Anteil des Gesamtbedarfs des Kunden i ,
der vom Standort j geliefert wird, $j \in N, i \in M$

- Zielfunktion:

$$\text{Minimiere } Z_{\text{WLP}} = \sum_{j \in N} f_j y_j + \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} \quad (9)$$

so dass

- Restriktionen:

$$\sum_{j \in N} x_{ij} = 1 \quad \text{für alle } i \in M \quad (10)$$

$$x_{ij} \leq y_j \quad \text{für alle } i \in M, j \in N \quad (11)$$

$$x_{ij} \geq 0 \quad \text{für alle } i \in M, j \in N, y_j \in \{0,1\} \text{ für alle } j \in N \quad (12)$$

Die Zielfunktion (9) minimiert die Summe aus Fix- und Transportkosten.

Restriktion (10) sichert, dass jeder Kunde seinen Bedarf erhält und Restriktion (11) bedeutet, dass wenn Standort j nicht gewählt wird ($y_j = 0$), ihm keine Kunden zugeordnet werden dürfen ($x_{ij} = 0$).

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Relaxation

Optimierungsproblem P: $\max_{x \in S_P} z_P(x)$ (13)

mit: $z_P(x)$ zu maximierende Zielfunktion des Problems P
 S_P Zulässigkeitsbereich – Menge aller zulässigen Lösungen – des Problems P

(Betrachtungen bei einem Minimierungsproblem P entsprechend!)

Relaxation R des Problems P: $\max_{x \in S_R} z_R(x)$

mit: $S_R \supseteq S_P$ Zulässigkeitsbereich von R ist eine Obermenge des Zulässigkeitsbereich von P

und: $z_R(x) \geq z_P(x)$ für alle $x \in S_P$
Für alle zulässigen Lösungen des Originalproblems P ist der Zielfunktionswert des relaxierten Problems eine obere Schranke für den Zielfunktionswert.

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Idee: Das relaxierte Problem sollte einfacher lösbar sein als das Originalproblem.

Anwendungsbereich:

- Zur Bewertung der Qualität bekannter zulässiger Lösungen
- Zur Steuerung der Lösungssuche in Heuristiken
- Bei Schnittebenen- und Spaltengenerierungsverfahren, die auf LP beruhen
- Zum Abschneiden von Teilbäumen in Branch-and-Bound /Price/ Cut-Verfahren
(wird auch Bounding genannt)

Einfachste Methode zur Erzeugung von Relaxationen

- Weglassen einer Teilmenge von Restriktionen (LP-Relaxation eines MP, Basis für Schnittebenen und Branch-and-Cut-Verfahren), Ganzzahligkeitsrestriktionen weglassen
- Restriktionen nicht weglassen, sondern deren Verletzung mit “Kosten“ bestrafen (Grundidee der Lagrange Relaxation)

LP – Relaxation

LP – Relaxation eines MIP: Ganzzahligkeitsforderungen für die Variablen (die dies betrifft) werden weggelassen.
ZF-Wert des relaxierten Problems liefert bei Maximierung eine obere Schranke für den ZF-Wert des Originalproblems.

- Liefert die LP-Relaxation eine gute (obere) Schranke?
- Antwort hängt vom Modell ab und kann nicht allgemein formuliert werden.

Spezialfall: Optimale Lösung des Originalproblems ist ganzzahlig, ohne dass man Ganzzahligkeit explizit fordern muss.
→ LP-Relaxation lösen!
(gilt für viele Netzflußprobleme, Transportprobleme mit ganzzahligen Daten)

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Für spezielle andere Modelle, weiß man welche Formulierungen (warum) günstig sind.

Beispiel: Betrachten des Modells des FLP (WLP) (10), (11), (12)

Die Restriktionen (12) haben die Form

$$x_{ij} \leq y_j \text{ für alle } j \in N \text{ und } i \in M \text{ (} m \cdot n \text{ Restriktionen)}$$

Betrachtet man diese Restriktion aus der Sicht $x_{ij} > 0$ sei gegeben, d.h. der Kunde i ist dem Standort j zugeordnet, so muss der Standort j geöffnet sein, $y_j = 1$. Damit muss ein Standort geöffnet sein, wenn ihm mindestens ein Kunde zugeordnet ist. Dann werden durch diesen Standort höchstens $m = |M|$ Kunden beliefert.

→ aggregierte Darstellung:
$$\sum_{i \in M} x_{ij} \leq m \cdot y_j \quad \text{für alle } j \in N \quad (14)$$

(Durch Addition der Ungleichungen (12) erhält man (14), d.h. wenn eine Lösung (12) erfüllt, erfüllt sie auch (14).)

Verwendet man (14) anstelle (12) hat man “nur“ $n = |N|$ Restriktionen.

Ist die Forderung aber vorteilhaft?

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

LP-Relaxation des ursprünglichen Modells bedeutet:

$$0 \leq y_j \leq 1 \quad \text{für alle } j \in N$$

- Bei $f_j > 0$ für $j \in N$ gilt bei der aggregierten Formulierung für jede Lösung:

$$y_j = \frac{1}{m} \cdot \sum_{i \in M} x_{ij} \quad \text{für alle } j \in N \quad (15)$$

(aggregierte Restriktionen sind für LP-Relaxation aktiv.)

- In der ursprünglichen (disaggregierten) Formulierung wird y_j von demjenigen Kunden bestimmt, der den größten Anteil seines Bedarfs am betrachteten Standort deckt.

$$y_j = \max_{i \in M} x_{ij} \quad \text{für alle } j \in N \quad (16)$$

Man kann zeigen, dass der Zulässigkeitsbereich der disaggregierten Formulierung des Modells eine echte Teilmenge des Zulässigkeitsbereiches der aggregierten Formulierung darstellt. Damit sind die Lösungen der disaggregierten Formulierung i.a. näher an der ganzzahligen Lösung, weshalb sie vorzuziehen ist, obwohl man wesentlich mehr Restriktionen hat.

Lagrange Relaxation (Grundlegende Überlegungen)

Grundidee: Modelle besitzen Teilstrukturen für die es effiziente Algorithmen gibt

- Diejenigen Restriktionen im Modell identifizieren, die eine “gutartige Modellstruktur” verhindern.
- Die Verletzung dieser Restriktionen als “Strafkosten” in eine somit veränderte Zielfunktion verschieben.

Sehr hohe Strafkosten sollen Einhaltung der Restriktion erzwingen.

Wie kann man sinnvoll “Strafkosten” einführen, die die Zielfunktion nicht völlig verzerren? (“Sehr hohe” Strafkosten erzwingen Zulässigkeit, verändern aber die inhaltliche Bedeutung der Zielfunktion.)

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Formalisierung

MIP: maximiere $z_{\text{MIP}} = c^T x$
 so dass

$A^1 x \leq b^1$ m_1 “schwierige“ Restriktionen

$A^2 x \leq b^2$ m_2 “einfache“ Restriktionen

$x \in \mathbb{R}^n \cap X$

$m = m_1 + m_2$ Restriktionen, n Variablen

x beinhaltet sowohl ganzzahlige als auch reellwertige Variablen $x \in \mathbb{R}^n$. Mit X legt man fest, welche Variablen ganzzahlig bzw. reellwertig sind.

$A^1 x \leq b^1$ “schwierige“ Restriktionen \rightarrow Aus der Restriktionsmenge entfernen

$A^2 x \leq b^2$ “gutartige“ Restriktionen \rightarrow Mit diesen Restriktionen sollte das Problem einfach (aber nicht “zu einfach“) zu lösen sein.

Das neu entstehende Problem sollte eine gute Approximation des ursprünglichen Problems bezüglich des Zielfunktionswertes sein.

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Die Verletzung der schwierigen Restriktionen wird mit variablen Strafkosten bewertet:

Lagrange Multiplikatoren $\pi_i \geq 0$ für $i = 1, \dots, m_1$ Restriktionen

$$\pi^T = (\pi_1, \dots, \pi_{m_1})$$

$a_i^T x \leq b_i$ ist genau dann verletzt, wenn

$$a_i^T x > b_i$$

bzw. $b_i - a_i^T x < 0$ ist. (a_i^T : i -ter Zeilenvektor von A^1)

Bestrafung der Unzulässigkeit durch Addition des Terms:

$$\pi_i (b_i - a_i^T x) \text{ zur Zielfunktion}$$

→ Neues Modell

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Lagrange – Relaxation bzgl. der Restriktion $A^1x \leq b^1$

Maximiere $Z_{LR}(\pi) = c^T x + \pi^T \cdot (b^1 - A^1 x)$

so dass

$$A^2 x \leq b^2 \quad (m_2 \text{ einfache Restriktionen})$$

$$x \in \mathbb{R}^n \cap X$$

- Umfasst die Lösungsmenge des ursprünglichen Problems
- ZF-Wert und $Z_{LR}(\pi)$ ist für jeden Vektor $\pi \geq 0$ eine obere Schranke für den Optimalwert des MIP. (Für jede zulässige Lösung gilt: $\pi^T \cdot (b^1 - A^1 x) \geq 0$)

Veränderte Schreibweise der ZF:

Maximiere $Z_{LR}(\pi) = (c^T - \pi^T A^1)x + \pi^T b^1$

d.h. Für festes π hat man lediglich die ZF – Einträge verändert und muss das durch die “einfachen“ Restriktionen definierte Problem lösen.

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

Aber $Z_{LR}(\pi)$ hängt in nichtlinearer Weise von π ab. Man sucht ein $\bar{\pi}$, welches den ZF-Wert minimiert.

$$Z_{LD} = \min_{\pi \geq 0} Z_{LP}(\pi) \quad \text{Lagrange – duales Problem}$$

Beispiel: (nur zur Illustration der Definitionen):

$$\begin{array}{ll} \max: & Z_{SPK} = 3x_1 + 4x_2 + 2x_3 + 5x_4 \text{ so dass} \\ & \left. \begin{array}{l} x_1 + x_2 \leq 1 \\ x_1 + x_3 \leq 1 \\ x_1 + x_4 \leq 1 \\ x_2 + x_4 \leq 1 \end{array} \right\} \text{Set – Packing Problem} \\ & (x_1, x_2, x_3, x_4)^T \in X = \{0, 1\}^4 \end{array}$$

Alle Packing – Restriktionen werden als schwierige Restriktionen betrachtet

1.1 Standardprobleme der kombinatorischen Optimierung und Relaxationen

→ Lagrange – relaxiertes Problem

$$\begin{aligned} \text{Maximiere} \quad Z_{\text{LP-SPK}}(\pi_1, \pi_2, \pi_3, \pi_4) = & (3 - \pi_1 - \pi_2 - \pi_3) \cdot x_1 + (4 - \pi_1 - \pi_4) \cdot x_2 \\ & + (2 - \pi_2) \cdot x_3 + (5 - \pi_3 - \pi_4) \cdot x_4 + \pi_1 + \pi_2 + \pi_3 + \pi_4 \end{aligned}$$

$$\text{so dass } (x_1, x_2, x_3, x_4)^T \in X = \{0, 1\}^4$$

Es seien $\pi_1 = \pi_2 = \pi_3 = \pi_4 = 2$ beliebig gewählte Lagrange – Multiplikatoren.
Dann ist die optimale Lösung des Lagrange – relaxierte Problems: $x_1 = x_2 = x_3 = 0$,
 $x_4 = 1$. Der optimale Zielfunktionswert ist 9.

Bestimmung optimaler Lagrangemultiplikation: Siehe Kapitel 4.7.

1.2. Gemischt-ganzzahlige Optimierung und die Methode Branch and Bound

1.2.1. Beispiel eines gemischt-ganzzahligen LPs

Beispiel:

maximiere $z = x_1 + 4x_2$

so dass $5x_1 + 8x_2 \leq 40$

$-2x_1 + 3x_2 \leq 9$

$x_1, x_2 \geq 0$ und ganzzahlig

1. Initialisierung

P_1 ; LP Relaxation, aktuelle untere Schranke $z = -\infty$

opt. Lösung von P_1 : $x_1^0 = (1,55; 4,03)$, $z_1 = 17,68$

obere Schranke $\bar{z}_1 = z_1 = 17,68$

x_1^0 ist keine zulässige Lösung für das ganzzahlige Model. P_1 ist aktiv und wird Verzweigungsknoten.

1.2. Gemischt-ganzzahlige Optimierung und die Methode Branch and Bound

2. Verzweigung

Verzweigung zunächst nach x_1 (Prioritätsregel)

d.h. zusätzliche Restriktionen

$x_1 \leq 1$ (P_2 -Knoten) oder $x_1 \geq 2$ (P_3 -Knoten)

P_2 und P_3 heißen aktiv.

3. Lösung von P_2

opt. Lösung $x_2^0 = (1; 3,67)$, $z_2 = 15,67 = \bar{z}_2$

Da x_2^0 nicht ganzzahlig ist, heißt P_2 weiterhin aktiv.

→ nach LIFO-Regel

4. Knoten P_2 verzweigen

P_2 : $x_2 \leq 3$ (Knoten P_4), $x_2 \geq 4$ (Knoten P_5)

P_4 lösen: → ganzzahlige optimale Lösung von P_4

$x_4^0 = (1; 3)$, (für den Teilbaum unter P_4)

x_4^0 ist zulässig für das Problem P da ganzzahlig.

1.2. Gemischt-ganzzahlige Optimierung und die Methode Branch and Bound

P_4 : $x_1 \leq 1$, $x_2 \leq 3$ und andere Restriktionen

→ Knoten P_4 heißt terminiert (nicht mehr aktiv)

$\wedge z_4$ ist untere Schranke für den Optimalwert von P ,
(zulässige Lösung gefunden) $\underline{z} = z_4 = 13$

5. Backtracking über P_2 nach P_5

$x_2 \geq 4$ zum optimalen Tableau von P_2 hinzufügen

→ dualer Simplexschritt ist nicht möglich ($x_2 \geq 4 \wedge x_1 \leq 1$ kollidiert mit
 $-2x_1 + 3x_2 \leq 9$), d.h. P_5 besitzt keine zulässige Lösung

→ Backtracking über P_2 nach P_1 führt zur “Auswahl” von P_3

6. Lösung von P_3

optimale, nicht-ganzzahlige Lösung: $x_3^0 = (2; 3,75)$, $z_3 = 17 = \bar{z}_3$

Da $\bar{z}_3 = 17 > \underline{z} = 13 \rightarrow P_3$ ist weiter zu verzweigen

Kein Bounding!

1.2. Gemischt-ganzzahlige Optimierung und die Methode Branch and Bound

7. Verzweigung von P_3 nach x_2

→ Knoten P_6 ($x_2 \leq 3$) und Knoten P_7 ($x_2 \geq 4$)

opt. Lösung von P_6 : $x_6^0 = (3, 2; 3)$; $z_6 = 15,2 = \bar{z}_6$

(man beachte: x_1 ist nun wieder nicht-ganzzahlig)

8. LIFO-Strategie – von P_6 aus weiter verzweigen

→ Knoten P_8 ($x_1 \leq 3$) und Knoten P_9 ($x_1 \geq 4$)

P_8 besitzt eine optimale ganzzahlige Lösung: $x_8^0 = (3; 3)$ $z_8 = 15 = \bar{z}_8$

Damit ist eine bessere, insgesamt (also für P) zulässige Lösung gefunden worden, als wir bei P_4 hatten.

→ Aktualisierung der unteren Schranke gemäß $\underline{z} = z_8 = 15$

(P_8 liefert optimale Lösung – man weiß es nur noch nicht!)

1.2. Gemischt-ganzzahlige Optimierung und die Methode Branch and Bound

→ 9. Backtracking zu P_6 , von P_6 nach P_9

P_9 liefert einen optimalen ZF-Wert $z_9 = 14 \leq 15 = \underline{z}$
(schlechter als schon bekannte untere Schranke),
 P_9 wird terminiert.

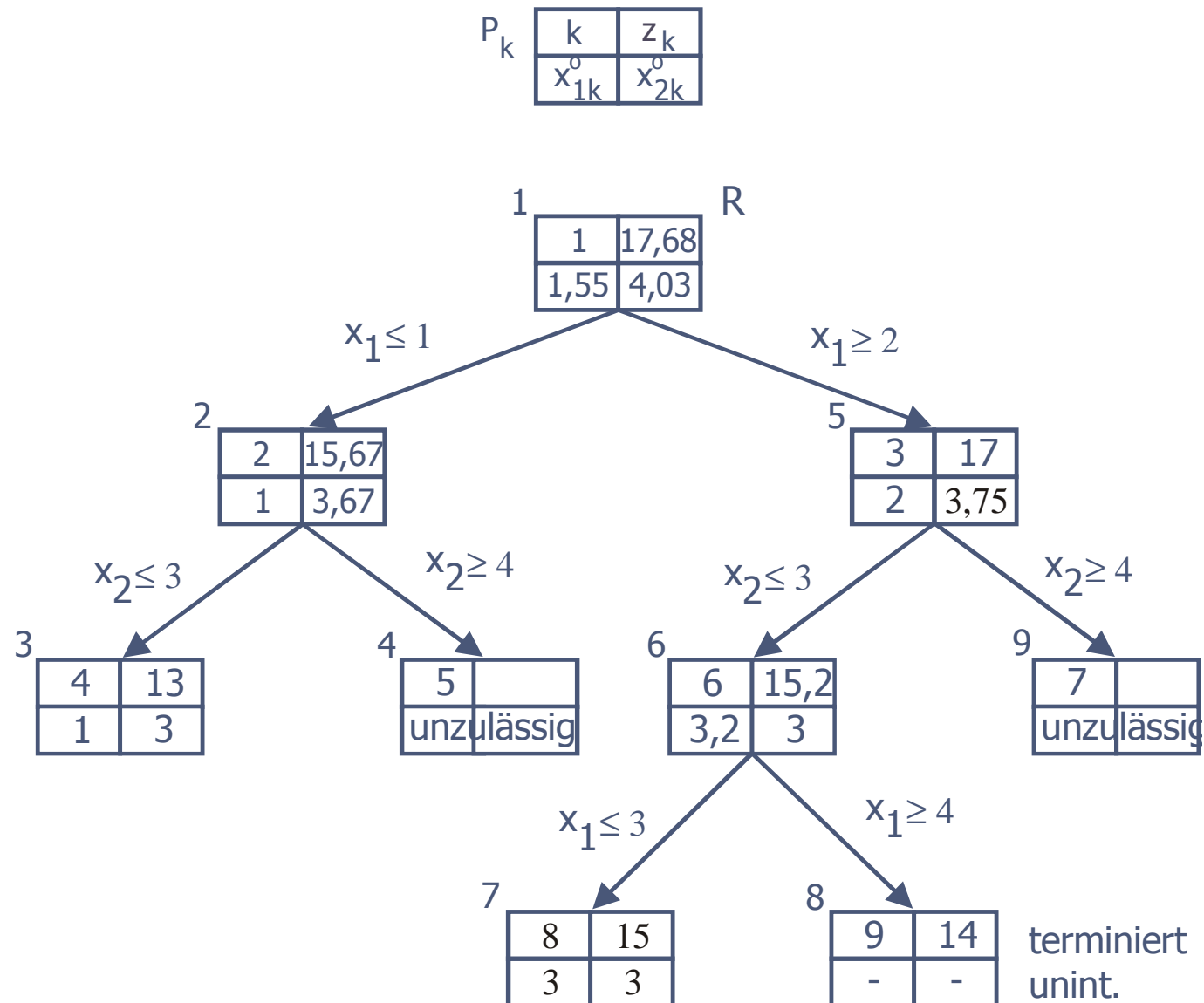
10. es gibt noch einen aktiven Knoten P_7

erreicht durch Backtracking von P_6 über P_3 nach P_7

P_7 besitzt keine zulässige Lösung.

Kein offener (aktiver) Knoten mehr.

→ Optimal ist die beste bekannte Lösung, d.h. die aktuelle untere
Schranke: Lösung von P_8



1.2. Gemischt-ganzzahlige Optimierung und die Methode Branch and Bound

1.2.2. Branch-and-Bound: Der Ansatz von Dakin

Problem:

$$\max z = c^T x$$

$$Ax \leq b \quad \text{LP-Relaxation}$$

$$x \geq 0$$

$$x_j \text{ ganzzahlig für } j = 1, \dots, p; x^T = (x_1, \dots, x_p, \dots, x_n)$$

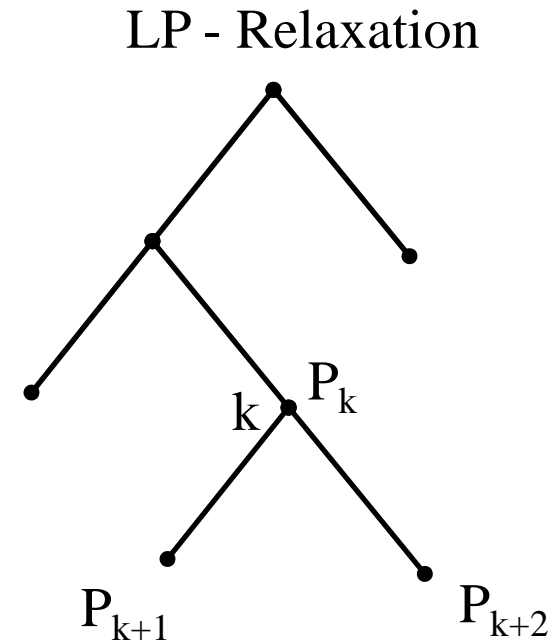
Gemischt-ganzzahliges Problem

Ansatz (Dakin 1965)

1. LP-Relaxation
2. Ist eine ganzzahlig geforderte Variable in einer optimalen Lösung der LP-Relaxation nicht ganzzahlig, dann
3. ist ihre Ganzzahligkeit in den optimalen Lösungen der Teilmodelle (Folgeknoten) durch geeignete ganzzahlige Schranken zu erreichen.

1.2. Gemischt-ganzzahlige Optimierung und die Methode Branch and Bound

$$\begin{aligned}\max z &= c^T x \\ Ax &\leq b \\ x &\geq 0\end{aligned}$$



Verletzung der Ganzzahligkeit

Problem P_k , repräsentiert von Knoten k

x_k^o optimale Lösung von k

Die Komponente $x_{k,j}^o$ ($1 \leq j \leq p$) von x_k^o erfülle die Ganzzahligkeitsbedingung nicht. (k wird unterdrückt)

d.h. $x_j^o = [x_j^o] + f_j$, $0 < f_j < 1$

1.2. Gemischt-ganzzahlige Optimierung und die Methode Branch and Bound

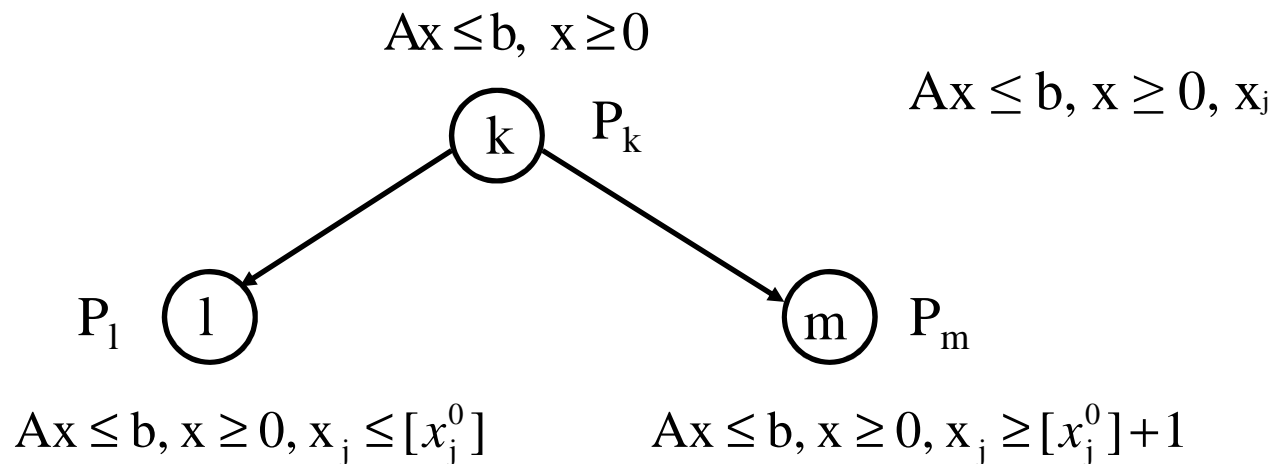
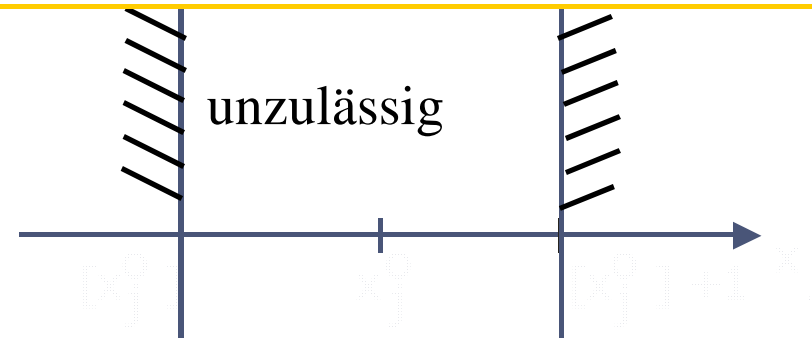
Verzweigen wie folgt:

→ obere Schranke $[x_j^0]$

hinzufügen

→ untere Schranke $[x_j^0]+1$

hinzufügen



Eigenschaften der Verzweigung

- In jedem Knoten wird nach einer Verzweigungsvariablen in zwei Teilmodelle verzweigt.

1.2. Gemischt-ganzzahlige Optimierung und die Methode Branch and Bound

- Damit entsteht eine Folge von Teilmodellen P_k , $k=1,2,\dots$
- Die Lösungsräume X_k für P_k sind LP-Relaxationen mit zusätzlich eingeführten ganzzahligen Bounds für einige oder alle der Variablen, für die Ganzzahligkeit gefordert ist.
- X_k sind dennoch Relaxationen bezüglich der Ganzzahligkeitsforderungen
→ Ihre optimalen ZF-Werte z_k^0 sind obere Schranken für die entsprechenden gemischt-ganzzahligen Teilmodelle

Technische Details

- Wenn das optimale Tableau für das vorangehende Modell (übergeordnete Bauebene) vorliegt:
→ analog zum Gomory-Verfahren vorgehen

1.2. Gemischt-ganzzahlige Optimierung und die Methode Branch and Bound

- Falls mehrere Variable Ganzzahligkeitsforderungen nicht erfüllen, Konfliktlösung nach folgenden Kriterien (nach welcher Variablen verzweigen?):
 1. die zu erwartende ZF-Wert-Verschlechterung (z.B. beim ersten dualen Schritt für jedes x_j)
 2. extern vorgegebene Prioritäten
 3. Abweichung einer Variablen vom nächsten ganzzahligen Wert (ganzzahlige Unzulässigkeit).

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Implizite Enumeration

Modell:

(1)

$$\begin{array}{ll} \max & Z = c^t x \\ \text{s.d.} & Ax \leq b \\ & x \in \{0, 1\} \end{array}$$

Annahme: $c \leq 0$, da jedes x_i mit $c_i > 0$ durch $x_i' = 1 - x_i$ ersetzt werden kann.

Zerlegung:

Es sei

$$(2) \quad L_0 = \{ x \mid Ax \leq b, x_i = 0 \vee x_i = 1 \text{ für alle } i \}$$

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

In Knoten K_k wird aufgespalten in

$$(3) \quad \begin{array}{l} L_{k+1} = L_k \cap \{ x \mid x_j = 0 \} \\ L_{k+2} = L_k \cap \{ x \mid x_j = 1 \} \end{array}$$

Ein x_j wird gewählt, welches auf dem Pfad zu K_k noch nicht zur Aufspaltung benutzt wurde.

wobei:

I_k : Indexmenge der bereits festgelegten Variablen

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

$$(4) \quad \begin{array}{l} L_k^1 = \{ j \mid j \in I_k \quad \text{und} \quad x_j = 1 \} \\ L_k^0 = \{ j \mid j \in I_k \quad \text{und} \quad x_j = 0 \} \\ N_k = \{ j \mid j \notin I_k \} \end{array}$$

⇒ Schranken können wie folgt festgelegt werden:

Folgendes Problem im Knoten K_k ($\sum_{j \in L_k^0} c_j x_j = 0$) wird betrachtet:

$$(5) \quad \begin{array}{l} \max Z_k = \sum_{j \in N_k} c_j x_j + \sum_{j \in L_k^1} c_j \\ \text{s.d.} \quad \sum_{j \in N_k} a_{ij} x_j \leq b_i - \sum_{j \in L_k^1} a_{ij} = S_i, \quad i = 1, \dots, m \\ x_j \in \{0, 1\}, \quad j \in N_k \end{array}$$

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Voraussetzung $c_j \leq 0$

\Rightarrow Maximum der ZF wird ohne Berücksichtigung der NB für $x^0(k)$ durch Festlegung von $x_j = 0 \quad \forall j \in N_k$ erreicht.

\Rightarrow **obere Schranke:**

$$(6) \quad \boxed{\bar{Z}_k = \sum_{j \in L_k^1} c_j}$$

und falls alle $S_i \geq 0 \Rightarrow x^0(k)$ zulässig und $\underline{Z}_k = \bar{Z}_k^0$

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Terminierung erfolgt, wenn

(7)

$\bar{Z}_k = \underline{Z}_k$	die optimale Lösung auf einem Zweig gefunden ist
<u>oder</u>	
$\bar{Z}_k \leq \underline{Z}_0$	der beste bisher gefundene zulässige ZF – Wert \underline{Z}_0 ist höher als der höchstens ab Knoten K_k noch erreichbare ZF - Wert

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Verzweigung:

Es sei $\bar{S}_k = \{ i \mid S_i < 0 \}$

- Falls $\bar{S}_k = \emptyset$

\Rightarrow terminiere K_k (da zulässige Lösung)

- Falls $\bar{S}_k \neq \emptyset$, so sei

$$(8) \quad \boxed{Q_k = \{ j \mid j \in N_k \text{ und } a_{ij} < 0 \text{ für einige } i \in \bar{S}_k \}}$$

Soll eine zulässige Lösung K_{k+1} erreicht werden:

\Rightarrow **mindestens eine Variable**, deren Index in Q_k enthalten ist,
muß den Wert 1 annehmen.

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Anwendung folgender Regel für Auswahl des x_j zur Erreichung
von Zulässigkeit:

(bei Verzweigung von x_j wobei $j \in Q_k$ und $x_j = 1$)

Es sei

(9)

$$U_k = \sum_{i=1}^m \max \{ 0, -S_i \} = - \sum_{i \in \bar{S}_k} S_i$$

die Unzulässigkeit von (5).

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Wählt man x_j , so ist beim nachfolgenden Knoten

$$U_{k+1}(j) = \sum_{i=1}^m \max \{ 0, -S_i + a_{ij} \}$$

⇒ Wähle x_j so, daß im nächsten Knoten die „Unzulässigkeit“
möglichst gering ist, d.h.:

$$U_{k+1}(p) = \min_{j \in Q_k} U_{k+1}(j)$$

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Beispiel:

$$\begin{aligned} \max Z &= -5x_1 - 7x_2 - 10x_3 - 3x_4 - x_5 \\ \text{s.d.} \quad &-x_1 + 3x_2 - 5x_3 - x_4 + 4x_5 \leq -2 \\ &2x_1 - 6x_2 + 3x_3 + 2x_4 - 2x_5 \leq 0 \\ &x_2 - 2x_3 + x_4 + x_5 \leq -1 \\ &x_j \in \{0, 1\}, j = 1, \dots, 5 \end{aligned}$$

Der jeweilige Pfad von K_0 bis K_k sei durch den Index – Vektor P_k bezeichnet, in dem die Elemente unterstrichen seien für die $x_j = 0$.

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Zunächst ist

$$N_0 = \{ 1, \dots, 5 \}, \quad L_k^1 = L_k^0 = \emptyset$$

$$\bar{Z}_0 = \infty$$

$$\underline{Z}_0 = -\infty$$

$$S = (-2, 0, -1)$$

$$U_0 = 3$$

$$Q_0 = \{ 1, 3, 4 \}$$

Für die Zerlegungsentscheidung werden verglichen:

$$U_1(1) = 4$$

$$U_1(3) = 3$$

$$U_1(4) = 5$$

mit der Konsequenz der Zerlegung nach x_3 .

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Am Knoten K_1 für $x_3 = 1$ ist dann:

$$\bar{Z}_1 = -10$$

$$S = (3, -3, 1)$$

$$Q_1 = \{ 2, 5 \}$$

$$U_2(2) = 0$$

$$U_1(5) = 2$$

Nächste Zerlegung nach x_2

Knoten K_2 :

$$P_2 = (3, 2)$$

$$\bar{Z}_2 = -17 = \underline{Z}_2 = \underline{Z}_0$$

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Da Zulässigkeit erreicht ist ($S = (0, 3, 0)$), erfolgt die Terminierung und Berechnung von

Knoten K_3

$$P_3 = (3, \underline{2})$$

$$S = (3, -3, 1)$$

$$Q_3 = \{5\}$$

Da $t_2 = \sum_{j \in N_k} \min \{ 0, a_{2j} \} > S_2$, d.h. $-2 > -3$,

kann keine Zulässigkeit erreicht werden

\Rightarrow Terminierung

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Knoten K_4 :

$$\begin{aligned} P_4 &= (\underline{3}) \\ S &= (-2, 0, -1) \\ Q_4 &= \{ 1, 4 \} \end{aligned}$$

Wieder ist $t_3 = 0 > S_3 = -1$

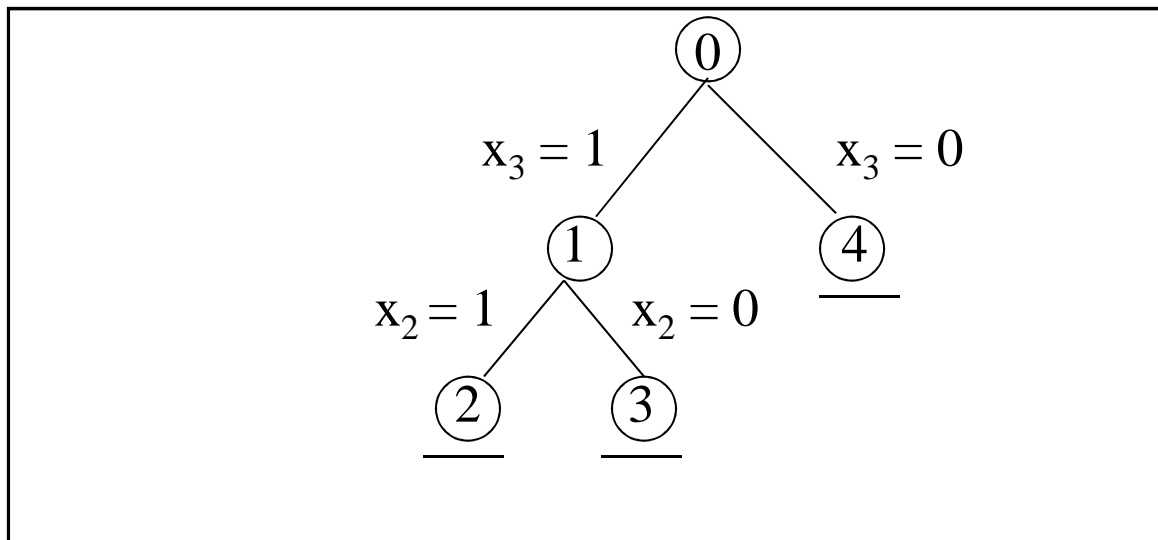
\Rightarrow Terminierung

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Optimale Lösung:

$$x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 0$$

$$Z = -17 \quad (\text{Knoten 2 mit } P = (3, 2))$$



1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Ersatznebenbedingungen (Surrogate Constraints)

- nach Festlegung gewisser Variablen bis zu einem Knoten ist häufig eine **zulässige Lösung** des 0/1 – Modells **nicht mehr möglich**
(siehe Knoten K_3 und K_4)

⇒ dies stellt man oft erst nach **vielen Schritten** bzw. nach Überprüfung **sämtlicher NB** fest

⇒ Die Ersatznebenbedingung erlaubt

die schnelle Feststellung, ob ein Lösungsraum, welcher Obermenge des betrachteten Lösungsraumes ist, leer ist und damit dieser Knoten terminiert werden kann.

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

In dem am Knoten K_k zu betrachtenden Problem (5) werden die NB

$$(10) \quad \sum_{j \in N_k} a_{ij} x_j \leq S_i, \quad i = 1, \dots, m$$

ersetzt durch die NB

$$(11) \quad \sum_{i=1}^m \sum_{j \in N_k} \alpha_i a_{ij} x_j \leq \sum_{i=1}^m \alpha_i S_i$$

mit $\alpha = (\alpha_1, \dots, \alpha_m) \geq 0$

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Für beliebige $\alpha \geq 0$ gilt

$$L_k(\alpha) \supseteq L_k,$$

wenn $L_k(\alpha)$ bzw. L_k die Lösungsräume von (11) bzw. (10) bezeichnen (d.h. also die jeweilige Menge zulässiger binärer Lösungen!)

Wenn gilt:

$$L_k(\alpha) = \emptyset$$

$$\Rightarrow \bullet L_k = \emptyset$$

- die Terminierung kann bereits aufgrund von (11) durchgeführt werden.

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Definition und Berechnung bester Ersatznebenbedingungen

Eine Terminierung durch Ersatznebenbedingungen sollte so früh wie möglich durchgeführt werden.

⇒ notwendig ist, daß sich $L_k(\alpha)$ möglichst wenig von L_k unterscheidet, d.h. die Bestimmung „optimaler“ α .

Es sei

$$(12) \quad g(\alpha) = \max_{x \in L_k(\alpha)} \sum_{j \in N_k} c_j x_j$$

Da $L_k(\alpha) \supseteq L_k$ ist auch $g(\alpha) \geq Z_k - \sum_{j \in L_k^1} c_j$ für alle $x \in L_k$ und $\alpha \geq 0$.

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Man bezeichnet

$$L_k(\alpha) \text{ stärker als } L_k(\alpha')$$

wenn gilt:

$$g(\alpha) < g(\alpha')$$

Damit könnte diejenige Ersatznebenbedingung als „optimal“ bezeichnet werden, für die gilt:

$$(13) \quad g(\alpha') = \min_{\alpha \geq 0} g(\alpha)$$

Zur Bestimmung der α' kann approximativ die Ganzzahligkeitsbedingung für die x_j zunächst fallen gelassen werden.

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Die NB wird als die beste bezeichnet, für die gilt:

$$(14) \quad g'(\alpha_0) = \min_{\alpha \geq 0} g'(\alpha)$$

mit

$$(15) \quad \begin{aligned} g'(\alpha) &= \max \sum_{j \in N_k} c_j x_j \\ \sum_{j \in N_k} \sum_{i=1}^m \alpha_i a_{ij} x_j &\leq \sum \alpha_i S_i \\ 0 \leq x_j &\leq 1, \quad j \in N_k \end{aligned}$$

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Zur numerischen Bestimmung von α^0 helfen folgende Betrachtungen:

Am Knoten K wird jetzt folgendes Modell betrachtet

(Verzicht auf die Ganzzahligkeitsbedingung):

$$(16) \quad \begin{array}{l} \max f_k = \sum_{j \in N_k} c_j x_j \\ \text{s.d. } \sum_{j \in N_k} a_{ij} x_j \leq S_i, \quad i = 1, \dots, m \\ 0 \leq x_j \leq 1, \quad j \in N_k \end{array}$$

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Das duale Problem lautet:

$$(17) \quad \begin{aligned} \min d_k &= \sum_{i=1}^m S_i v_i + \sum_{j \in N_k} w_j \\ \text{s.d. } \sum_{i=1}^m a_{ij} v_i + w_j &\geq c_j, \quad j \in N_k \\ v_i, w_j &\geq 0, \quad i = 1, \dots, m, j \in N_k \end{aligned}$$

• Falls (16) keine zulässige Lösung hat,

\Rightarrow existiert auch keine ganzzahlige Lösung und Knoten K kann terminiert werden.

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

- Gibt es eine Lösung für (16),

\Rightarrow dann existiert auch eine optimale Lösung für (17).

Gezeigt worden ist, daß die beste Ersatz – NB, d.h.

$$L(\alpha^0) \subseteq L(\alpha) \quad \forall \alpha \geq 0$$

die ist, für die gilt:

$$\alpha^0 = v^0$$

Ist nun

$$L_k(\alpha^0) = \emptyset$$

dann auch L_k und K_k kann terminiert werden.

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Beispiel:

$$\begin{array}{ll}\max & Z = -5x_1 - 7x_2 - 10x_3 - 3x_4 - x_5 \\ \text{s.d.} & -x_1 + 3x_2 - 5x_3 - x_4 + 4x_5 \leq -2 \\ & 2x_1 - 6x_2 + 3x_3 + 2x_4 - 2x_5 \leq 0 \\ & x_2 - 2x_3 + x_4 + x_5 \leq -1 \\ & x_j \in \{0, 1\}, j = 1, \dots, 5\end{array}$$

Da $\sum c_j = -26$ ist $\underline{Z}_0 = -26$

1.3. Branch and Bound für binäre Optimierungsprobleme: Implizite Enumeration

Löst man das entsprechende LP (16), so erhält man

$$x_0 = (0, 1/3, 2/3, 0, 0) \quad Z_0 = \bar{Z}_0 = -9$$

mit den entsprechenden Ersatzvariablen

$$v_1 = 0, \quad v_2 = 8/3, \quad v_3 = 9$$

Das ergibt als Ersatznebenbedingung:

$$\frac{16}{3}x_1 - 7x_2 - 10x_3 + \frac{43}{3}x_4 + \frac{11}{3}x_5 \leq -9$$

Aufgrund dieser NB kann x_3 auf 1 fixiert werden.

1.4. Allgemeine Darstellung der Methode Branch and Bound

Problem: $P: \max f(x), x \in S$

Anstelle ein Problem P zu lösen, konstruiert man eine Folge von Problemen P_k (mit Lösungsbereich X_k) mit der Zielstellung:

a) entweder optimale Lösung von P_k bestimmen

oder

b) zeigen, dass Optimalwert von P_k nicht besser als bester bisher bekannter ZF-Wert ist

oder

c) zeigen, dass X_k leer ist.

1.4. Allgemeine Darstellung der Methode Branch and Bound

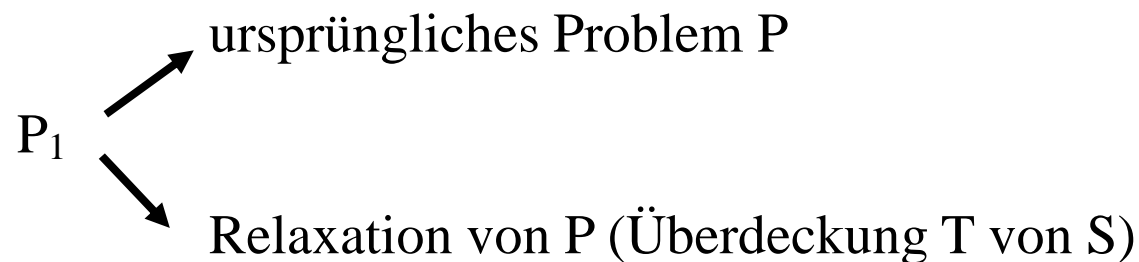
Lösungsaufwand verringern durch:

- geschicktes Verzweigen in Teilprobleme P_k (Branching!)
- Terminieren von Teilbäumen vor deren vollständiger Generierung nach b) oder c) (Bounding!)

Grundelemente des Branch-and-Bound-Verfahrens

1. Initialisierung

P_1 - Definition des Ausgangsmodells (Wurzelknoten)



1.4. Allgemeine Darstellung der Methode Branch and Bound

Maximierungsprobleme: aktuelle untere Schranke \underline{z}

Minimierungsprobleme: aktuelle obere Schranke angeben! \bar{z}

(bekannte zulässige Lösung benutzen oder $\underline{z} = -\infty$ bzw. $\bar{z} = +\infty$ setzen)

2. Verzweigung (Branching)

Bedeutet:

- Auswahl eines noch nicht untersuchten Teilmodells (Verzweigungsknoten) oder
- Erzeugung von Teilmodellen (Verzweigungsknoten) aus einem untersuchten aber nicht terminierten (Teil-)Modell.

1.4. Allgemeine Darstellung der Methode Branch and Bound

Verzweigung wird durch Regeln gesteuert:

Verzweigungsregeln definieren:

- wie aus einem Modell die Teilmodelle (Folgeknoten) generiert werden und
- welcher Knoten als nächster betrachtet (entwickelt) wird (Knotenauswahlregeln, Kontrollstrategien)

Beispiele für Knotenauswahlregeln:

1. Orientiert an den Bounds der aktiven (noch nicht überprüften) Knoten, z.B. Best-First Search (Bestensuche)
2. Extern festgelegte Regeln

1.4. Allgemeine Darstellung der Methode Branch and Bound

3. Schrankenunabhängige Regeln z.B. LIFO (Last In, First Out):
soweit möglich vom zuletzt erzeugten Knoten aus verzweigen;
bei Backtracking zum zuletzt generierten noch aktiven Knoten
gehen
(Depth-First, Tiefensuche)

Knotenauswahlregeln können dynamisch verändert werden:

z.B. zuerst: LIFO (um schnell \underline{z} bzw. \bar{z} zu bekommen).
danach: Schrankenabhängige Regel benutzen.

3. Terminierung (Bounding)

eines Knotens (Teilbaums) erfolgt, wenn entweder

- nachgewiesen ist, dass der Lösungsbereich X_k von Teilmodell P_k leer ist oder

1.4. Allgemeine Darstellung der Methode Branch and Bound

- eine obere Schranke für den ZF-Wert von P_k angegeben wird, \bar{z} , die nicht größer als die aktuelle untere Schranke \underline{z} des Ausgangsproblems ist (beim Max.problem).

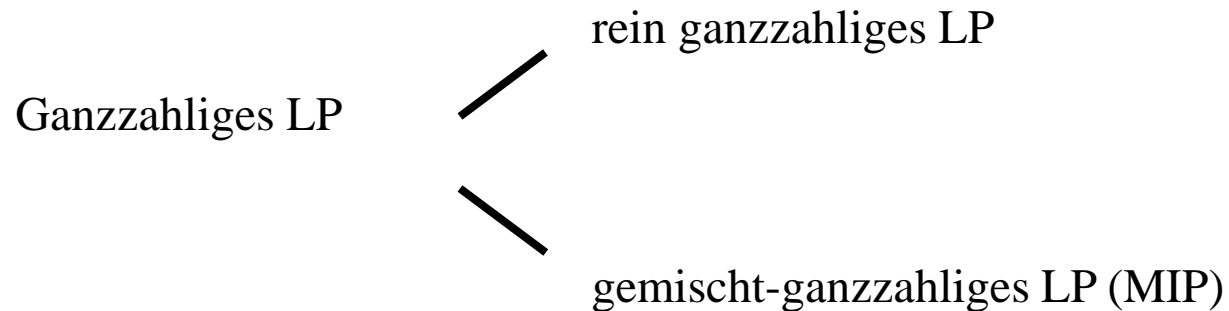
→ weitere Verzweigung des Knotens ist nicht mehr sinnvoll

Gesamt-Prinzip-Algorithmus \Rightarrow Struktogramm (Verzweigung in Teilprobleme P_k liege vor!)

<u>Initialisierung</u> Ausgangsproblem P_1 und bei <u>Maximierung</u> (Minimierung) aktuelle <u>untere</u> (obere) Schranke \underline{z} (\bar{z}). Knoten $k = 1$ heißt aktiv.	
Existieren aktive Knoten?	
ja	nein
Wähle Verzweigungsregel	
Wähle Knoten k (Teilproblem P_k) gemäß Verzweigungsregel	
Besitzt P_k eine zulässige Lösung?	
ja	nein
Bestimme optimale Lösung x_k^o von P_k und <u>obere</u> Schranke \bar{z}_k (untere Schranke \underline{z}_k)	
$\bar{z}_k > \underline{z}$ ($\underline{z}_k < \bar{z}$)	
ja	nein
Optimale Lösung x_k^o zulässig für Problem P ?	
ja	nein
Aktualisiere <u>untere</u> (obere) Schranke: $\underline{z} = \bar{z}_k = \underline{z}_k$ ($\bar{z} = \underline{z}_k = \bar{z}_k$) und aktuelle beste Lösung $x = x_k^o$	Knoten k heißt aktiv
Terminiere Knoten k und alle aktiven Knoten k' mit $\bar{z}_{k'} \leq \underline{z}$ ($\underline{z}_{k'} \geq \bar{z}$)	
Terminiere Knoten k	

Problemstellung, Beispiele

Forderung ganzzahliger Werte für gewisse (eventuell alle) Variablen.



Gründe:

- Keine beliebige Teilbarkeit der durch entsprechende EV modellierten Ressourcen (Menschen, Gebäude, Maschinen, Projekte etc.)

1.5 Ganzzahlige Lineare Optimierung: Das Schnittebenenverfahren von Gomory

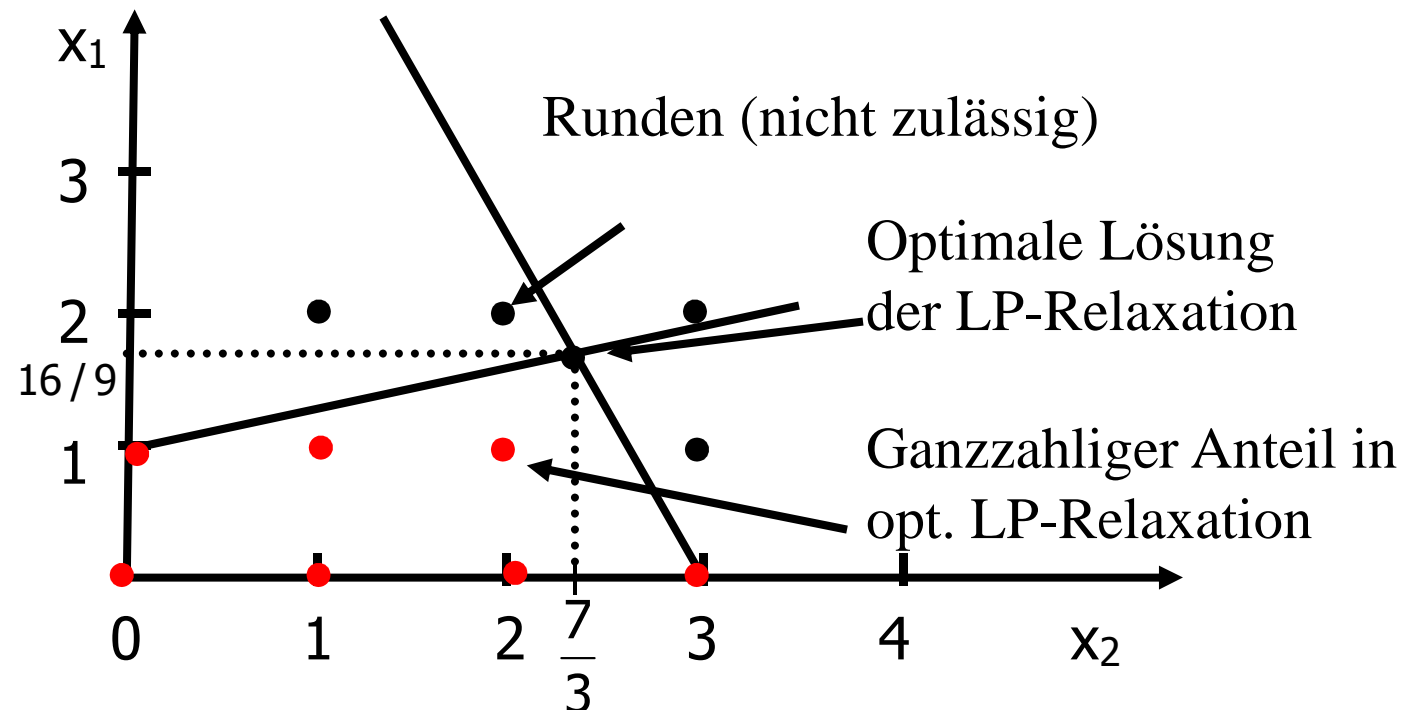
- Logische Bedingungen: „0-1 - Variablen“, z.B. Multiple-choice-Bedingungen, genau eine von n Alternativen wählen

$$\sum_{j=1}^n y_j = 1 \wedge y_j = \begin{cases} 1, & \text{falls } A_j \text{ gewählt} \\ 0, & \text{sonst} \end{cases}$$

Beispiel 1:

$$\begin{array}{ll} \text{maximiere} & z = 5x_1 + 3x_2 \\ \text{s.d.} & 3x_1 + 8x_2 \leq 24 \\ & 6x_1 - 2x_2 \leq 6 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \text{ ganzzahlig} \end{array}$$

1.5 Ganzzahlige Lineare Optimierung: Das Schnittebenenverfahren von Gomory



Es existieren 7 zulässige Lösungen. (rote Punkte im Bild)

Optimale Lösung LP-Relaxation: $\hat{x}_1 = \frac{16}{9}, \hat{x}_2 = \frac{7}{3}$

ganzzahl. Runden: $x_1=2, x_2=2$

ganzzahl. Anteil: $x_1=1, x_2=2$ optimale ganzzahlige Lösung

$$\hat{z}_{\text{Relax.}} = \frac{80}{9} + 7 \approx 16, \hat{z}_{\text{Ganzz.}} = \frac{10}{2} + 6 = 11$$

Das Schnittebenenverfahren von Gomory

Für rein ganzzahlige Probleme, d.h. alle Variablen sind ganzzahlig gefordert.

1. Schnittebenen:

Grundmodell:

$$\begin{array}{ll} \max & z = c^T x \quad x, c \in \mathbb{R}^n \\ \text{so dass} & Ax = b \quad b \in \mathbb{R}^m, A = A_{m,n} \\ & x \geq 0 \end{array}$$

(Schlupfvariablen ggf. schon in x enthalten.)

Eine primal und dual zulässige Basis liege vor.

O.E.d.A. Indizierung so, dass

- die ersten m Variablen $i=1, \dots, m$ die BV sind,
- die letzten $j=m+1, \dots, n$ Variablen die NBV sind, $n > m$.

Basisdarstellung der i -ten BV: $x_{B_i} = b_i^* - \sum_{j=m+1}^n a_{ij}^* x_j$ (1)

($i=1, \dots, m$)

1.5 Ganzzahlige Lineare Optimierung: Das Schnittebenenverfahren von Gomory

- Aufspaltung nichtganzzahliger b_i^* und a_{ij}^* in ganzzahligen Anteil und nichtnegativ gebrochenen Anteil.

$$\left. \begin{aligned} b_i^* &= [b_i^*] + h_i \\ a_{ij}^* &= [a_{ij}^*] + h_{ij} \\ [b_i^*], [a_{ij}^*] &\in \mathbb{Z}, 0 \leq h_i, h_{ij} < 1 \end{aligned} \right\} \quad (2) \quad \text{z. B.} \quad \begin{aligned} b_i^* &= 1,732 \\ &= [1,732] + h_i \\ &= 1 + 0,732 \end{aligned}$$

- Einsetzen von (2) in (1):

$$\begin{aligned} x_{B_i} &= [b_i^*] + h_i - \sum_{j=m+1}^n ([a_{ij}^*] + h_{ij}) x_j \\ &= \underbrace{[b_i^*] - \sum_{j=m+1}^n [a_{ij}^*] x_j}_{(I)} + \underbrace{h_i - \sum_{j=m+1}^n h_{ij} x_j}_{(II)} \quad (3) \end{aligned}$$

Nach Def. ganzzahlig \longrightarrow Teil (II) ganzzahlig ist NHB für Ganzzahligkeit von x_{B_i}

Herleitung der Gomory-Restriktionen

Es gilt: $h_{ij} \cdot x_j \geq 0$, da $h_{ij} \geq 0 \wedge x_j \geq 0$. $\Rightarrow \sum_{j=m+1}^n h_{ij} x_j \geq 0, d.h. \quad - \sum_{j=m+1}^n h_{ij} x_j \leq 0$

$$\Rightarrow h_i - \sum_{j=m+1}^n h_{ij} x_j \leq h_i < 1 \quad (4)$$

$$\Rightarrow h_i - \sum_{j=m+1}^n h_{ij} x_j \leq 0 \quad (5)$$

$$\left[\Leftrightarrow - \sum_{j=m+1}^n h_{ij} x_j \leq -h_i \right]$$

ist eine notwendige Bedingung für die Ganzzahligkeit von II.

Fügt man eine Gomory-Variable als Schlupfvariable hinzu, entsteht die Gomory-Restriktion

$$x_G - \sum_{j=m+1}^n h_{ij} x_j = -h_i \quad (6) \quad (\text{Schnittebene})$$

1.5 Ganzzahlige Lineare Optimierung: Das Schnittebenenverfahren von Gomory

Da in BL (2) alle NBV $j=m+1, \dots, n$ gleich 0 sind, gilt in (6): $x_G = -h_i$.

Modifikation der Simplexmethode

- Dem optimalen Simplextableau des relaxierten LP die Gomory-Restriktion hinzufügen!
 - Duale Zulässigkeit der BL bleibt bestehen.
 - Wegen $-h_i < 0$ wird primale Zulässigkeit verletzt.
- Deshalb:
Eine duale Simplexiteration mit Gomory-Restriktion als Pivotzeile durchführen.

Es ergibt sich:

1. Die primale Unzulässigkeit der hinzugefügten Zeile wird beseitigt.
 2. Die Gomory-Variable wird NBV.
- Ist duale Simplexiteration nicht möglich, so ist Lösungsraum leer.
Im kontinuierlichen Lösungsraum gibt es keine ganzzahligen Lösungen.

Schnittebenen-Verfahren von Gomory: Der Algorithmus

- 1. Schritt:** Löse das Grundmodell (LP-Relaxation) mit der Simplexmethode
- 2. Schritt:** Sind in der optimalen BL alle Ganzzahligkeitsbedingungen erfüllt, ist opt. ganzzahlige Lösung gefunden (STOPP), sonst:
- 3. Schritt:** Spalte die Komponenten b_i^* der rechten Seite (d.h. der BL) im optimalen Tableau für die ganzzahlig geforderten Variablen gemäß (2) auf. Der Zeilenindex t , für den eine Gomory-Restriktion gebildet wird, wird aus
$$h_i = \max_i \{h_i \mid h_i \text{ aus Aufspaltung (2)}\}$$
 ermittelt. Ist t nicht eindeutig, wähle kleinsten derartigen Zeilenindex.

4. Schritt: Spalte die b_i der t-Zeile des Tableaus nach (2) auf:

$$a_{tj}^* = [a_{tj}^*] + h_{tj} \quad , a_{tj}^* \text{ ganzzahlig}, 0 \leq h_{tj} < 1$$

Bilde die Gomory-Restriktion nach (6).

5. Schritt: Füge Gomory-Restriktion dem aktuellen Tableau hinzu und iteriere mit der dualen Simplexmethode. Ist derartige Iteration nicht möglich, dann gibt es keine ganzzahlige Lösung (STOPP), sonst gehe zu Schritt (2).

1.5 Ganzzahlige Lineare Optimierung: Das Schnittebenenverfahren von Gomory

Beispiel (LP aus Beispiel 1):

1. Schritt: Das optimale Endtableau der LP-Relaxation ist:

		C _j					
		5	3				
C _{bi}	X _{Bi} \ X _j	X ₁	X ₂	X ₃	X ₄	b _i [*]	h _i
3	X ₂		1	1/9	-1/18	7/3	1/3
5	X ₁	1		1/27	4/27	16/9	7/9 ← t
	ΔZ _j			14/27	31/54	143/9	

Optimales Tableau für relaxiertes Problem

2. Schritt: x₁ und x₂ sind nicht ganzzahlig.

3. Schritt:
$$h_t = \max \left\{ \frac{1}{3}, \frac{7}{9} \right\} = \frac{7}{9}, \quad t=2$$

1.5 Ganzzahlige Lineare Optimierung: Das Schnittebenenverfahren von Gomory

4. Schritt:
$$x_{G1} - \frac{1}{27}x_3 - \frac{4}{27}x_4 = -\frac{7}{9}$$

5. Schritt: Das ergänzte Tableau lautet:

		C _j					
		5	3				
C _{bi}	X _j X _{Bi}	X ₁	X ₂	X _{G1}	X ₃	X ₄	b _i *
3	X ₂		1		1/9	-1/18	7/3
5	X ₁	1			1/27	4/27	16/9
	X _{G1}		1		-1/27	-4/27	-7/9
	ΔZ _j				14/27	31/54	143/9

1.5 Ganzzahlige Lineare Optimierung: Das Schnittebenenverfahren von Gomory

Nach einem dualen Simplex-Schritt erhält man:

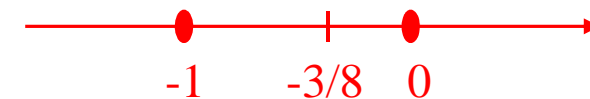
		C _j						
		5	3					
C _{bi}	X _j X _{Bi}	X ₁	X ₂	X _{G1}	X ₃	X ₄	b _i [*]	h _i
3	X ₂		1	-3/8	1/8		21/8	5/8
5	X ₁	1		1	0		1	0
	X ₄			-27/4	1/4	1	21/4	-
	ΔZ _j			31/8	3/8		103/8	

2. Schritt: x_2 ist nicht ganzzahlig.

3. Schritt: $h_t = \max \left\{ \frac{5}{8} \right\} = \frac{5}{8}, t=1$

4. Schritt: $x_{G2} - \frac{5}{8}x_{G1} - \frac{1}{8}x_3 = -\frac{5}{8}$

$$-3/8 = -1 + 5/8$$



1.5 Ganzzahlige Lineare Optimierung: Das Schnittebenenverfahren von Gomory

5. Schritt: Das weiterhin ergänzte Tableau ist:

	C_j	5	3					
C_{bi}	X_j X_{Bi}	x_1	x_2	x_{G1}	x_{G2}	x_3	x_4	b_i^*
3	x_2		1	-3/8		1/8		21/8
5	x_1	1		1		0		1
	x_4			-27/4		1/4	1	21/4
	x_{G2}			-5/8	1	-1/8		-5/8
	Δz_j			31/8		3/8		103/8

1.5 Ganzzahlige Lineare Optimierung: Das Schnittebenenverfahren von Gomory

Eine weitere duale Simplex-Iteration liefert das optimale Endtableau des ganzzahligen Problems:

		C_j	5 3						
C_{Bi}	X_j	X_{Bi}	x_1	x_2	x_{G1}	x_{G2}	x_3	x_4	b_i^*
3	x_2			1	-1	1			2
5	x_1		1		1	0			1
	x_4				-8	2		1	4
	x_3				5	-8	1		5
	Δz_j				2	3			11

2. Schritt: x_1 und x_2 sind ganzzahlig. Stopp!

1.6. Ausblick: Branch-and-Cut-Verfahren (BaC)

- BaC – Algorithmen sind spezielle BaB – Algorithmen: In jedem Knoten des BaB Entscheidungsbaumes wird ein Schnittebenenverfahren angewendet. Obwohl beide Verfahren (BaB- und Schnittebenen-Verfahren) exakte Verfahren sind, macht es Sinn, sie zu kombinieren. Vorteile sind:

1. In manchen Fällen finden Schnittebenenverfahren keine optimale Lösung:

- Menge der bekannten Schnittebenen reicht nicht aus um eine ganzzahlige Lösung zu finden
 - Für eine Gruppe von Schnittebenen gibt es kein effizientes Separationsverfahren
- Zerlegung in Teilprobleme (**Branching**) hilft in solchen Fällen.

1.6 Ausblick: Branch-and-Cut-Verfahren (BaC)

2. Schnittebenenverfahren erlauben oft die Berechnung einer sehr guten unteren Schranke bezüglich der optimalen Lösung → **Bounding** hält die Entscheidungsbäume oft klein.
3. Es ist oft effizienter das Generieren gültiger Ungleichungen von einem bestimmten Punkt an zu beenden und statt dessen zu verzweigen.
4. Man kann gültige Ungleichungen, die in einem Knoten des BaB – Baumes generiert werden unter Umständen in anderen Knoten weiter verwenden.

1.6 Ausblick: Branch-and-Cut-Verfahren (BaC)

Generisches Schnittebenenverfahren

- 1: Löse die LP-Relaxation des Modells
- 2: Solange das Abbruchkriterium nicht erfüllt ist, wiederhole:
- 3: Löse das Separationsproblem für eine oder mehrere Klassen von Ungleichungen heuristisch oder exakt optimal
- 4: Füge die erzeugten Ungleichungen (oder eine Teilmenge dieser) zur LP – Relaxation hinzu.
- 5: Reoptimiere die LP – Relaxation mit dem dualen Simplexalgorithmus.

Dieses Schnittebenenverfahren wird in jedem Knoten des BaC – Algorithmus durchgeführt.

1.6 Ausblick: Branch-and-Cut-Verfahren (BaC)

Nach Abbruch des Schnittebenenverfahrens gilt:

- Entweder
- (1) es wurde eine zulässige Lösung gefunden, oder
 - (2) gezeigt, dass das Problem unzulässig ist (keine zulässige Lösung besitzt), oder
 - (3) eine bekannte obere Schranke überschritten, oder
 - (4) eine zulässige Lösung für die Relaxation bestimmt, die nicht zulässig für das ursprüngliche Problem ist (also eine untere Schranke darstellt).

- (1), (2), (3) → Terminierung des Knotens des BaC – Baumes
- (4) → weitere Verzweigung durchführen

Die gefundenen Schnittebenen werden in einem Cut – Pool dynamisch verwaltet.

2. Heuristiken und Metaheuristiken

- Heuristiken* sind Algorithmen, die ein gegebenes Optimierungsproblem mit akzeptablen Aufwand in möglichst guter Qualität lösen.

*in Zusammenhang mit Optimierungsproblemen auch: Approximationsalgorithmen. Die Künstliche Intelligenz verwendet eine viel umfassendere Definition zu Heuristiken (für Problemlösung)

Warum Heuristiken?

- Die zur Verfügung stehende Zeit reicht für eine exakte Problemlösung nicht aus.
- Der notwendige Speicherbedarf übertrifft den zur Verfügung stehenden Speicherbedarf
- Die Akzeptanz von Heuristiken kann größer sein, wenn diese einfacher verständlich sind als exakt optimierende Algorithmen.
- Heuristiken können bzgl. Hard- und Software (Implementierungsaufwand) kostengünstiger sein.

2. Heuristiken und Metaheuristiken

Hier nicht behandelt werden:

-Eigenschaften von Heuristiken:

- Rechen- und Speicheraufwand
- Qualität der Lösungen:
 - Bewertung von Heuristiken: Abhängigkeit von der Instanz
 - Worst-Case Analyse
 - Average-Case (Wahrscheinlichkeitsverteilung auf der Menge der Instanzen)
- Stopp – Regeln
- Eröffnungs- bzw. Verbesserungsverfahren
- [Metaheuristiken](#) sind übergeordnete Algorithmen die die Lösungssuche eines oder mehrerer abhängiger Algorithmen steuern. Sie beruhen auf einer Menge von Strategien die unabhängig vom zugrundeliegenden Problem und den abhängigen gesteuerten Algorithmen sind.

2.1 Greedy – Algorithmen

- Greedy Heuristiken sind Eröffnungsverfahren zur Konstruktion einer zulässigen Lösung des Optimierungsproblems

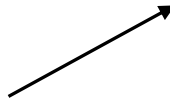
“greedy“ : gierig

Idee: In jedem Schritt eine Variable des Optimierungsproblems auf den Wert zu fixieren, der in diesem Schritt zur größten Verbesserung der Zielfunktion führt.


Formalisierung

Statische Unterteilung der Variablen in abhängige und unabhängige Variable

Verwendung eines anderen effektiveren Algorithmus in Abhängigkeit von den Werten der unabhängigen Variablen



Werte mit Hilfe des Greedy Algorithmus ermitteln



2.1. Greedy - Algorithmen

Festlegung der unabhängigen Variablen

Indexmenge $I = \{1, \dots, n\}$ der unabhängigen Variablen

Der Greedy – Algorithmus partitioniert die Menge I in die zwei Mengen:

$I^{\text{fix}} \subseteq I$ - Menge der fixierten Variablen

$I^{\text{frei}} \subseteq I$ - Menge der freien Variablen

In jedem Schritt des Algorithmus gilt:

$$I^{\text{fix}} \cup I^{\text{frei}} = I \quad \text{und}$$

$$I^{\text{fix}} \cap I^{\text{frei}} = \emptyset$$

- Am Anfang sind alle Variablen frei: $I^{\text{frei}} = I$, $I^{\text{fix}} = \emptyset$
- In jedem Iterationsschritt wird genau eine Variable aus der Menge I^{frei} entfernt und zur Menge I^{fix} hinzugefügt (eine Variable wird fixiert)
- Die Fixierung einiger Variablen schränkt den zulässigen Wert für die noch freien Variablen ein d. h.

sei $X_j(\bar{x}_k, k \in I^{\text{fix}})$ die Menge der mögliche (zulässigen) Werte der Variablen x_j , $j \in I^{\text{frei}}$, nachdem die x_k , $k \in I^{\text{fix}}$ auf $x_k = \bar{x}_k$ fixiert werden

2.1. Greedy - Algorithmen

Auswahl der nächsten zu fixierende Variablen:

Lösen eines Hilfsproblems P_j für jede freie Variable $x_j, j \in I^{\text{frei}}$.

P_j : Maximiere $z_j = \bar{c}(x_j)$
 so dass $x_j \in X_j(\bar{x}_k, k \in I^{\text{fix}})$

P_j : “Optimalität“ bzgl. einer Hilfs-Zielfunktion $c(\bar{x}_j)$ die für jeden möglichen Wert von x_j eine Bewertung vornimmt
 Optimaler Wert von x_j unter der Bedingung, dass die fixierten Variablen ihre Werte beibehalten. Der neue Wert der Variablen gehört zum eingeschränkten Zulässigkeitsbereich $X_j(\bar{x}_k, k \in I^{\text{fix}})$.

\Rightarrow Der Algorithmus generiert eine zulässige Lösung

Man fixiert die Variable x_j , deren Bewertung z_j am größten ist.

$$j^* = \arg \max_{j \in I^{\text{frei}}} \{z_j\}$$

Man setzt $x_{j^*} = \bar{x}_{j^*}$. Falls alle Variable fixiert ($\in I^{\text{fix}}$) sind, terminiert der Algorithmus.

Beispiele: Rucksackproblem, Zuordnungsproblem, TSP (nächster Nachbar)
→ Übungen

2.2 Lokale Suche (Verbesserungsverfahren)

Idee: Eine gegebene zulässige Lösung durch eine definierte Menge elementarer Operationen in eine andere zulässige Lösung so transformieren, dass diese neue Lösung eine bessere Bewertung besitzt. Findet man eine solche Lösung, wiederholt man das beschriebene Vorgehen. Kann man mit den definierten elementaren Operationen keine “bessere” Lösung finden, terminiert das Verfahren in einem lokalen Optimum.

Nachbarschaftsbegriff $N(x)$: Optimierungsproblem: minimiere $z = c(x)$
so dass $x \in X \subseteq \mathbb{R}^n$

Nachbarschaft $N(x) \subset X$ einer Lösung $x \in X$ ist die Menge aller derjenigen Lösungen, die mit Hilfe von definierten elementaren Operationen aus x generierbar sind.

2.2. Lokale Suche (Verbesserungsverfahren)

Der Simplexalgorithmus ist ein sehr bekanntes lokales Suchverfahren. Dabei ist die Elementaroperation ein zulässiger Basistausch

→ d.h. lokale Suchverfahren müssen keine Heuristiken sein, sind es aber häufig. Beispiele in 3.2. (k-opt Verfahren beim TSP und VRP)

Formale Darstellung

Es sei x^0 eine bekannte zulässige Lösung des Optimierungsproblem. Die lokale Suche tauscht in jedem Iterationsschritt $t \in \{0, 1, \dots\}$ die aktuelle Lösung x^t gegen eine bezüglich der Bewertung c bessere Lösung x^{t+1} aus, wobei gilt:

$$x^{t+1} \in N^t(x^t) \subset X \text{ und } c(x^{t+1}) < c(x^t)$$

Das Verfahren terminiert, falls in einem Iterationsschritt t keine “benachbarte“ Lösung existiert, die besser als die aktuelle Lösung ist, d.h. es gilt:

$$c(x') \geq c(x^t) \quad \text{für alle } x' \in N^t(x^t)$$

- Suche bis erste verbessernde Lösung in $N^t(x^t)$ gefunden ist: [Erstensuche](#)
- Beste Lösung in $N^t(x^t)$ ermitteln: [Bestensuche](#)
- [k-Erstensuche](#) (k verbessernde Lösungen aus $N^t(x^t)$, davon die beste wählen)

2.3. Simulated Annealing

Idee: “Simuliertes Ausglühen“: z.B. bei Metallen.

Das Material wird aufgewärmt und geht dann beim Abkühlen von seiner flüssigen Form in eine kristalline Struktur über. Dabei ist das Ziel den Prozess des Abkühlens so zu gestalten, dass gleichmäßige Kristallstrukturen entstehen. → Langsames Abkühlen ist notwendig.

Motivation eines physikalischen Modells für das Abkühlen:

- X sei die Menge aller Zustände des Materials, $j \in X$ ein spezieller Zustand.
(Ein Zustand repräsentiert eine mögliche Kristallstruktur.)
- Jeder Zustand $j \in X$ besitzt ein Energieniveau E_j .
- Bei einer Temperatur T kann ein Zustandsübergang von $i \in X$ zu einem “Nachbarzustand“ $j \in X$ (geringe Veränderung) stattfinden.

Die Wahrscheinlichkeit eines solchen Übergangs hängt von den Energieniveaus E_i und E_j ab.

2.3. Simulated Annealing

Akzeptanzkriterium

Ist die Energie des Nachbarzustandes $j \in X$ kleiner als die des aktuellen Zustands $i \in X$, (d.h. $E_j < E_i$), findet der Übergang mit Sicherheit statt. Ist aber

$E_j > E_i$, wird der Übergang mit Wahrscheinlichkeit

$$P(E_j, E_i, T) = e^{-\frac{E_j - E_i}{kT}} \quad (k - \text{ Boltzmann-Konstante})$$

akzeptiert, ansonsten verbleibt das System (Material) im Zustand i .

(Metropolis, 1953) Akzeptanzkriterium

Folgerung: Die Wahrscheinlichkeit eines Zustandsübergangs von i nach j nimmt mit steigender Energiedifferenz $E_j - E_i$ ab.

→ Metropolis Algorithmus zur Simulation des thermischen Gleichgewichts eines Materials bei gegebener Temperatur

2.3. Simulated Annealing

Metropolis Algorithmus

- 1: Setze die Temperatur T auf einen festen Wert und wähle zufällig einen aktuellen Zustand $i \in X$.
- 2: Wiederhole, bis ein Stoppkriterium erfüllt ist:
- 3: Bestimme zufällig einen Nachbarzustand $j \in X$ („kleine“ Veränderung von i)
- 4: Berechne die Energiedifferenz $\Delta E = E_j - E_i$
- 5: Falls $\Delta E < 0$, dann setze $i := j$
(Akzeptanz des neuen Zustands j)
- 6: Andernfalls setze $i := j$ mit Wahrscheinlichkeit
 $P^{\text{acc}}(\Delta E, T) := e^{-\Delta E/kT}$ (bedingte Akzeptanz)

2.3. Simulated Annealing

Man kann beweisen:

- Bei “genügend langer“ Ausführung des Algorithmus und Wahl einer geeignet gewählten “kleinen Veränderung“ der Zustände wird eine stationäre Verteilung entstehen

d.h. Die Zustände werden mit einer bestimmten Wahrscheinlichkeit angenommen, die unabhängig von der Anzahl der Iterationen des Algorithmus ist

$$P(X = i) = \frac{1}{Z(T)} e^{-\frac{E_i}{kT}}$$

→ **Boltzmann Verteilung:**

mit der Normierungskonstanten $Z(T) = \sum_{i \in X} e^{-\frac{E_i}{kT}}$

Eigenschaften der Verteilung:

Geht $T \rightarrow \infty$ (sehr hoher Temperatur), ist jeder Zustand gleich wahrscheinlich

$$P(X=i) \rightarrow \frac{1}{|X|} \text{ für } T \rightarrow \infty$$

Bei sehr kleinen Temperaturen, $T \rightarrow 0$, werden ausschließlich Zustände mit minimaler Energie angenommen.

2.3. Simulated Annealing

$$P(X=i) \rightarrow \begin{cases} 0 & , \text{ falls } i \notin X_{\min} \\ \frac{1}{|X_{\min}|} & , \text{ falls } i \in X_{\min} \end{cases} \quad \text{für } T \rightarrow 0$$

Dabei ist X_{\min} die Menge der Zustände mit minimaler Energie, d.h.

$$X_{\min} = \{ i \in X \mid \text{Es existiert kein } j \in X \text{ mit } E_j < E_i \}$$

Zusammenfassung:

- Im Zustand $i \in X$ wird zufällig ein Nachbarzustand ausgewählt (“geringe“ Modifikation des Ausgangszustands)
- Nachbarzustand wird immer akzeptiert, wenn er geringere Energie aufweist. Ist die Energie größer wird der Nachbarzustand mit einer Wahrscheinlichkeit akzeptiert die mit steigender Energiezunahme sinkt.
- Bei niedriger Temperatur erreicht man nach einer großen Zahl von Iterationen die Zustände minimaler Energie

? Analogie zwischen diesem Abkühlungsprozeß und Minimierungsproblemen !

2.3. Simulated Annealing

Analogien Abkühlungsprozeß - Minimierungsproblem

Abkühlung	Minimierungs- problem	<u>Kommentar:</u>
Zustand	Zulässige Lösung	-Es gibt keine natürliche Analogie zu dem Produkt kT aus Boltzmann Konstante und Temperatur
Energie	Zielfunktionswert	→ Deshalb führt man einen künstlichen Steuerparameter T ein, den man auch Temperatur nennt.
Zustandsübergang	Lösung aus der Nachbarschaft	
Temperatur	Steuerungsparameter	
Fester Zustand	Letzte gefundene zulässige Lösung	Kühlt man zu schnell ab, werden aufgrund des Akzeptanzkriteriums nahezu nur Nachbarzustände mit niedriger Energie akzeptiert. Dem entspricht die Forderung nach Verbesserung in jedem Schritt bei der lokalen Suche.
Schnelles Abkühlen	Lokale Suche →	

2.3. Simulated Annealing

→ Simulated Annealing ist eine zufallsgesteuerte lokale Suche mit einem modifizierten (verallgemeinerten) Akzeptanzkriterium

(Die Nachbarschaft wird aber nicht explizit durchsucht, sondern es wird zufällig eine benachbarte Lösung (als Stichprobe) bestimmt.)

Mit welcher Temperatur T initialisiert man den Algorithmus?
Wie verringert man diese Temperatur von Iteration zu Iteration?

~ Abkühlungs-
strategie

Mögliche Abkühlungsstrategien:

1. Setze T_0 (Initialisierung) auf den Wert der maximalen ZF – Differenz zweier Nachbarlösungen
2. Führe für jede Temperatur T_t , $t = 0, 1, \dots, K$ eine festgelegte Anzahl von n_t Iterationen durch
3. Verringere die Temperatur in jedem n_t -ten Iterationsschritt gemäß $T_{t+1} := \alpha \cdot T_t$, $\alpha \in [0.8, 0.99]$
4. Terminiere, falls in den letzten N Iterationen keine Verbesserung aufgetreten ist oder T_t einen vorgegebenen Wert unterschritten hat.

2.3. Simulated Annealing

Generischer Simulated-Annealing-Algorithmus (SAA) für die Lösung eines Minimierungsproblems

- 1: (Initialisierung) Initialisiere die Temperatur T_0 , die Anzahl n_0 der Schritte bei konstanter Temperatur und die Startlösung $x \in X$.
- 2: Setze $t := 0$
- 3: (Abkühl Schleife) Wiederhole:
- 4: (Schleife bei konstanter Temperatur) Wiederhole n_t – mal:
- 5: (Zufällige Nachbarlösung) Bestimme zufällige Nachbarlösung $x' \in N(x)$
- 6: (Zielfunktionsänderung) Berechne die Zielfunktionsdifferenz
$$\Delta c = c(x') - c(x)$$
- 7: (Akzeptanzkriterium) Falls $\Delta c < 0$, setze $x := x'$. Andernfalls bestimme eine Zufallszahl $z \in [0, 1]$. Falls $z < e^{-\Delta c / T_t}$, setze $x := x'$.
- 8: (Update) Bestimme eine neue Temperatur $T_{t+1} = f(t, T_t)$
- 9: Bestimme n_{t+1} und setze $t := t + 1$
- 10: bis Abbruchkriterium erfüllt ist.

-
- Unter schwachen Voraussetzungen an die Nachbarschaft und Abkühlstrategie kann man zeigen, dass der SAA mit Wahrscheinlichkeit 1 gegen eine optimale Lösung konvergiert (Beweise mit Hilfe der Theorie der Markov-Ketten)
 - Häufig werden Veränderungen am Schema des SAA vorgenommen um die Performance zu verbessern

2.4. Tabu Search

Idee: Auf der lokalen Suche aufbauende Metaheuristik

Wie steuert man die Suche zum Verlassen lokaler Optima?

- Simulated – Annealing verwendet den Zufall um lokale Optima (Minima) zu verlassen
- Tabu – Search greift auf Informationen über den bisherigen Suchprozess zurück und speichert diese in verschiedenen „Gedächtnissen“ (Speicher)

2.4. Tabu Search

Gedächtnis:

- Attribute bisher untersuchter Lösungen (bzw. des Übergangs von einer Lösung zu einer anderen) in einem “attributiven Speicher“ verwalten.
Vor dem Wechsel zu einer Lösung aus der Nachbarschaft nachsehen (in diesem Speicher) ob die Attribute der “neuen“ Lösung darauf hindeuten, dass diese Lösung schon vorher “besucht“ oder der Übergang bereits durchgeführt wurde.
→ Wenn ja, dann wurde diese Lösung wahrscheinlich schon untersucht, sie wird deshalb als “tabu“ erklärt. Im anderen Fall wird der Übergang erlaubt.
- Attributive Speicher enthalten nur einzelne Attribute einer Lösung oder eines Lösungsübergangs.
 - Vorteil: Kleiner Speicherbedarf
 - Nachteil: Mehrere Lösungen oder Lösungsübergänge können gleiche Attribute aufweisen. Damit können Lösungen tabu sein, die bisher nicht untersucht wurden.

2.4. Tabu Search

Beispiel: Binäres Optimierungsproblem mit n Entscheidungsvariablen
(z.B. Rucksack- oder Set-Covering-Problem)

Eine Lösung x wird durch Angabe der Werte der EV in diesem Vektor $x_T = (x_1, \dots, x_n)$ repräsentiert.

Beispiel für eine Nachbarschaft: Werte von zwei EV x_i und x_j , $i \neq j$

Invertieren: z.B: $x_i = 0$ und $x_j = 1$ wird zu
 $x_i' = 1$ und $x_j' = 0$

Mögliche Attribute zur Charakterisierung des Übergangs von einer Lösung zur anderen sind:

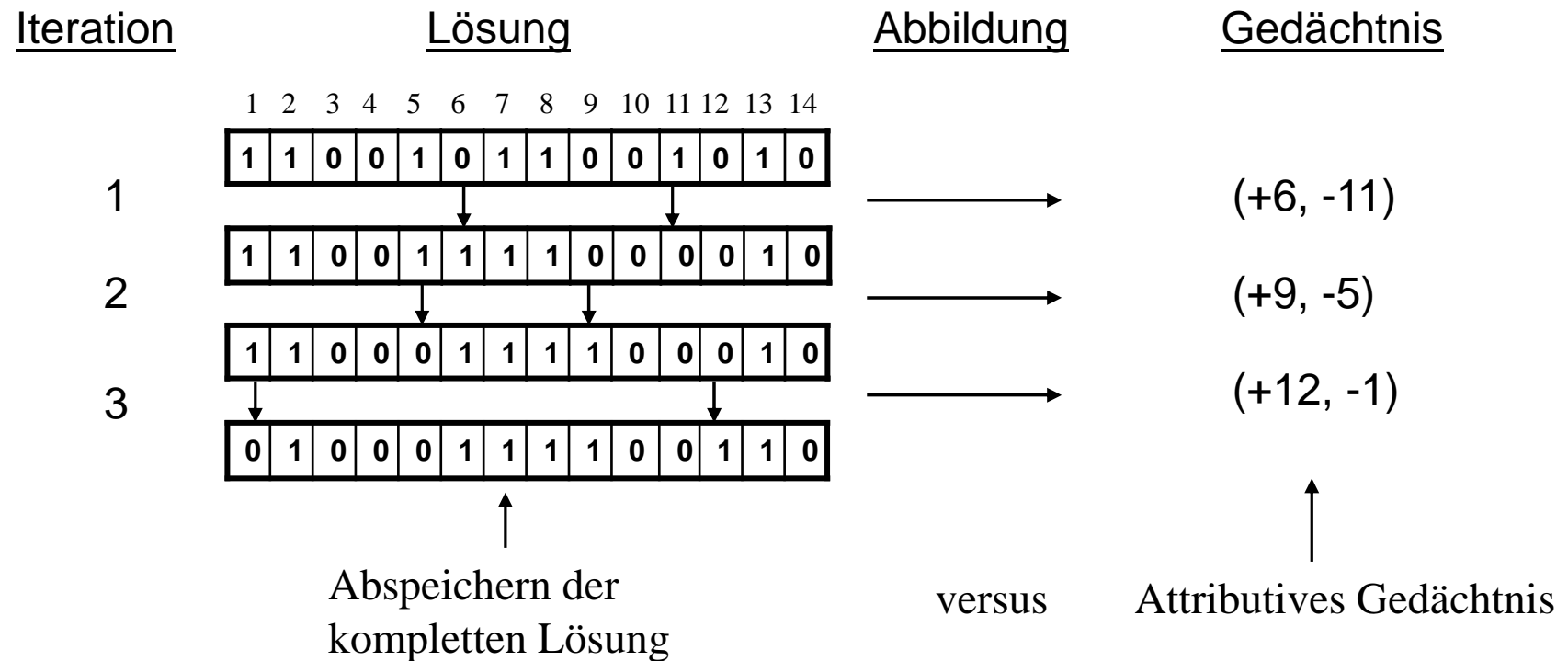
Inc (i)	: Index i der Variablen x_i , deren Wert erhöht wird.
Dec (j)	: Index j der Variablen x_j , deren Wert verkleinert wird.
Swap (+i, -j)	: Indizes i, j der Variablen, deren Wert erhöht bzw. verkleinert wird
$c(x') - c(x)$: Veränderung der ZF beim Übergang $x \rightarrow x'$
$g(x') - g(x)$: Veränderung einer problemspezifischen Funktion beim Übergang $x \rightarrow x'$

2.4. Tabu Search

Fortsetzung des Beispiels:

Benutzung des Attributs Swap (+i, -j) zum Speichern des bisherigen Lösungsprozesses.

3 Iterationen eines lokalen Suchverfahrens für den binären Vektor x und die Entwicklung des “Gedächtnisses” werden dargestellt.



2.4. Tabu Search

Zurück zur allgemeinen Betrachtung zur Tabu Search

Wie kann man mit Hilfe des attributiven Gedächtnisses erreichen, dass im Suchprozeß keine Zyklen entstehen?

Tabu – Restriktionen

- Eine Tabu – Restriktion geht aus von:

- aktueller Lösung x^t
- einer benachbarten Lösung $x' \in N^t(x^t)$
- dem aktuellen Status des attributiven Gedächtnisses

und bestimmt davon ausgehend

- ob die Nachbarlösung x' im aktuellen Iterationsschritt t tabu ist

Ist die Nachbarlösung tabu, darf sie im nächsten Schritt nicht gewählt werden. (Ausnahmen!). (Solche Übergänge zu Nachbarlösungen verbieten, die die Inversion eines aktuell gespeicherten Attribut zur Folge hatten.)

2.4. Tabu Search

Fortsetzung des Beispiels

Mögliche Tabu-Restriktionen die auf Attributen basieren

Nummer	Basis-Attribut	abgeleitete Tabu-Restriktion
1	Inc(i)	Verbiete das Verringern von x_i auf 0
2	Dec(j)	Verbiete das Erhöhen von x_j auf 1
3	Inc(i) und Dec(j)	Verbiete Schritte bei denen entweder 1 oder 2 auftritt
4	Swap (+i, -j)	Verbiete den (umgekehrten) Schritt Swap (+j, -i)
5	$c(x') - c(x)$	Verbiete Schritte mit einer ZF-Änderung von $c(x) - c(x')$
6	$g(x') - g(x)$	Verbiete Schritte, die zu einer Änderung der Funktion g um den Wert $g(x) - g(x')$ führen

Tabu Restriktion 3 ist restriktiver als 1 bzw. 2, dagegen ist 4 weniger restriktiv als 1 bzw. 2 bzw. 3

2.4. Tabu Search

Einfluss der Tabu-Restriktionen auf den Tabu-Status von Nachbarlösungen

(Gedächtnisspeicher aus Tabelle S. 62)

Die Nachbarlösungen im 4. Iterationsschritt seien bzgl. ihres Tabu-Status zu bewerten

Welche Übergänge zu einer Nachbarlösung sind tabu?

Tabu - Restriktion	verbotene Übergänge
1	Setzen der Variablen x_6, x_9 bzw. x_{12} auf 0
2	Setzen der Variablen x_{11}, x_5 bzw. x_1 auf 1
3	Setzen der Variablen x_6, x_9, x_{12} auf 0 <u>oder</u> Setzen der Variablen x_{11}, x_5, x_1 auf 1
4	Gleichzeitiges Setzen von $x_6 = 0$ und $x_{12} = 1$ <u>oder</u> $x_9 = 0$ und $x_5 = 1$ <u>oder</u> $x_1 = 0$ und $x_{12} = 1$

2.4. Tabu Search

Allgemein: Beschränkung auf das Kurzzeitgedächtnis um lange Tabu Listen zu vermeiden.

-Kurzzeitgedächtnis: Speichert die Attribute der zuletzt untersuchten Lösungen der Tabu Liste



der Länge k: Enthält Attribute der k letzten Lösungen (bzw. Lösungsübergänge)

Befinden sich beim Hinzufügen bereits k Attribute in der Tabu-Liste (als FIFO-Schlange organisiert), so wird das erste Attribut “vergessen”.

-Anspruchskriterien: Ist ein Kriterium, das den Tabu-Status bestimmter
(aspiration levels) Tabu-Restriktionen aufhebt

2.4. Tabu Search

Typische Beispiele für Anspruchskriterien in Tabu Search

Anspruchskriterium	Bedeutung
Anspruch in “Ermangelung“ (aspiration by default)	Falls alle Nachbarlösungen tabu sind (Ermangelung von nicht-tabu Alternativen), wird diejenige gewählt deren Tabu (bzgl. eines Maßes) „am wenigsten“ tabu ist
Anspruch durch eine “Zielfunktion“ (aspiration by objective)	Falls der ZF-Wert besser ist, als der bisher beste Wert, wird das “tabu“ der Lösung aufgehoben. “Besser“ kann sich auch auf die aktuell untersuchte Region des Lösungsraum beziehen.
Anspruch durch “Suchrichtung“ (aspiration by search direction)	Falls das zur Restriktion führende Attribut in einer Verbesserungsphase des Suchprozess gesetzt wurde und der Schritt zur Nachbarlösung auch verbessernd (aber tabu) ist, wird der Tabu-Status aufgehoben.

2.4. Tabu Search

Basialgorithmus: Tabu-Search mit Kurzzeitgedächtnis

- 1: (Initialisierung) Initialisiere das Verfahren mit einer zulässigen Lösung x^0
 und setze den Iterationszähler $t:=0$
- 2: (Iterationsschleife) Wiederhole;
- 3: (Bestensuche) Durchsuche die gesamte Nachbarschaft $N^t(x^t)$ mit
 Hilfe der definierten Elementaroperationen
- 4: (Auswahl) Wähle eine bzgl. einer Bewertungsfunktion beste
 Lösung $x' \in N^t(x^t)$, die nicht tabu ist, oder tabu ist und ein
 Anspruchsniveau erfüllt
- 5: (Iteration) Setze $x^{t+1} := x'$, $t:=t+1$, und aktualisiere das Kurzzeit-
 gedächtnis
- 6: bis ein Stopp-Kriterium erfüllt ist.

Erweiterungen: Häufigkeitsgedächtnis, Intensivierung, Verlassen des
 Zulässigkeitsbereichs – Strafkosten Diversifikation,
 Kandidatenlisten

2.5. Genetische und Evolutionäre Algorithmen (GEA)

Nachbildung von Gedanken der natürlichen Evolution insbesondere Charles Darwins “Survival of the fittest“ mit Hilfe der Operatoren: Selektion, Rekombination (Kreuzung) und Mutation

Die durchschnittliche Fitness einer Population von Individuen wird dadurch erhöht, dass sich Individuen mit überdurchschnittlicher Fitness bevorzugt vermehren und ihre Eigenschaften an ihre Nachkommen weitergeben.

Analogiebildung zur Nutzung dieser Theorie für die Lösung von Optimierungsproblemen

Zulässige Lösungen des Optimierungs- ↔ Individuen einer Population problems

Kodierte Lösung ↔ Chromosom (*Individuen werden durch Gene beschrieben. Die Gesamtheit der Gene bildet das Chromosom*)

Bewertung einer Lösung (*Verletzung von Restriktionen über Strafterme in die Zielfunktion integrieren*) ↔ Fitness des Individuums (*Zu maximierende Zielfunktion wird hier als Fitnessfunktion bezeichnet*)

2.5. Genetische und Evolutionäre Algorithmen (GEA)

Wie geben Individuen Teile ihrer Erbinformation (ihrer Chromosomen) an ihre Nachkommen weiter? [2 Modelle](#)

└→ **Genetische Algorithmen (GA):**

Zwei Individuen (Eltern) werden selektiert. Diese erzeugen durch [Rekombination](#) zwei Nachkommen (Kinder) und geben dabei Teile ihrer Erbinformationen weiter ([Kreuzungoperator](#), crossover). Zusätzlich werden gegebenenfalls einige Gene der so erzeugten Kinder durch [Mutation](#) zufällig verändert.

└→ **Evolutionstrategien (EA):**

Die Elternpopulation erzeugt aufgrund von Mutationen eine Menge von Nachkommen. Die Eltern können Teil der neuen Population sein oder werden direkt durch die Kinder ersetzt.

GA und EA - Unterscheidung ist historisch bedingt. Durch Weiterentwicklungen ist die Grenze fließend → nachfolgend: GEA (keine explizite Unterscheidung)

2.5. Genetische und Evolutionäre Algorithmen (GEA)

Algorithmus GEA

- 1: Initialisiere mit einer Population von zulässigen Lösungen x^1, \dots, x^p
- 2: Wiederhole;
- 3: Bestimme die [Fitness](#) aller Lösungen x^1, \dots, x^p der Population
- 4: Wähle p (nicht notwendig verschiedene) Lösungen x_s^1, \dots, x_s^p aus der Population mit Hilfe des [Selektionsoperators](#)
- 5: Erzeuge aus den p selektiven Lösungen p neue Lösungen x_n^1, \dots, x_n^p mit Hilfe des [Kreuzungsoperators](#)
- 6: Modifiziere die erzeugten Lösungen x_n^1, \dots, x_n^p mit Hilfe des [Mutationsoperators](#).
Bezeichne die mutierten Lösungen mit x^1, \dots, x^p
- 7: bis ein Stopp-Kriterium erfüllt ist.

2.5. Genetische und Evolutionäre Algorithmen (GEA)

=> Man muss folgende Module spezifizieren:

- Kodierung einer Lösung als Chromosom
- Populationsgröße
- Erzeugung einer Ausgangspopulation
- Selektionsoperator
- Kreuzungsoperator
- Mutationsoperator
- Stopp - Kriterium

Wichtigste Modellierungsentscheidung: Kodierung der Lösung z.B.

Binärkodierung: Die interessierenden Variablenwerte werden hintereinander als Binärkode in einen Vektor eingetragen

Binärkodierung und einfacher Kreuzungsoperator (single-point crossover)

1.Schritt: Bestimmung der Kreuzungsposition

Elternteil x^1 : (1 1 0 | 1 0 1 1 0)

Elternteil x^2 : (0 1 1 | 1 0 0 0 1)

2.Schritt: Verbinden der Elemente vor und nach der Kreuzungsposition

Kind x_n^1 : (1 1 0 | 1 0 0 0 1)

Kind x_n^2 : (0 1 1 | 1 0 1 1 0)

2.5. Genetische und Evolutionäre Algorithmen (GEA)

Bestimmung der Populationsgröße

Tradeoff zwischen

- Aufwand für Speicherung und Durchführung einer Iteration
- und
- erforderliche (minimale) Größe zur Nachbildung evolutionärer Prozesse

Bei vielen Anwendungen liegt die Populationsgröße zwischen 50 und 500 Individuen

Wichtige Aspekte bei der Generierung der Ausgangspopulation

- Individuen der Ausgangspopulation sollten zulässig sein
- Gegebenenfalls Eröffnungsheuristiken benutzen. Parametervariationen zur Generierung verschiedener Lösungen ausnutzen
- Ausreichende Heterogenität der Ausgangspopulation anstreben.

Selektion der Eltern

Roulette – Wheel – Operator: Am Häufigsten verwendeter Selektionsoperator

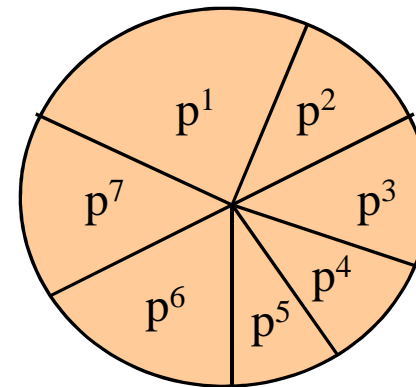
1. Berechnung der Summe der Fitness-Bewertungen über alle Individuen der Population.
2. Einem Individuum wird genau die Wahrscheinlichkeit für Selektion zugeordnet die seinem Anteil an der Gesamtfitness entspricht.

x^i : Fitness $f^i := f(x^i)$

Wahrscheinlichkeit p^i für Selektion:

$$p^i = \frac{f^i}{\sum_{j=1}^p f^j}$$

Individuen mit hoher Fitness haben
“große“, die mit geringer Fitness
“kleine“ Chancen als Eltern aus-
gewählt zu werden.



Populationsgröße = 7:
Drehen des Roulette Rades

2.5. Genetische und Evolutionäre Algorithmen (GEA)

Beispiel Set-Covering Problem

Matrix mit Kosteneinträgen:

	4	3	10	7	5	5	4	9	4
	1	0	1	1	0	0	1	1	0
	0	1	1	0	0	1	0	1	1
	1	0	0	0	1	1	0	0	1
	1	0	1	0	1	0	0	1	1
	0	0	1	1	1	0	0	0	0
	0	1	1	0	0	0	1	1	0
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9

Kodierung:

n-elementiger binärer Vektor (n: Anzahl der Spalten =9)

An j-ter Stelle steht eine 1, falls die Spalte in der Lösung enthalten ist (d.h. ausgewählt wurde), sonst steht eine 0.

Ein Vektor kann dann einer unzulässigen Lösung entsprechen, wenn nicht alle Zeilen durch die ausgewählten Spalten abgedeckt sind.

z.B. (0 1 0 0 1 1 0 0 1) Spalten 2, 5, 6, 9 gewählt

→ Zeile 1 ist nicht abgedeckt !!

2.5. Genetische und Evolutionäre Algorithmen (GEA)

Ausweg: Nicht abgedeckte Zeilen mit Strafkosten belegen

z.B. Konstante $Q = 20$ pro nicht abgedeckter Zeile in der ZF addieren

Fitness-Funktion: Angelehnt an die um Strafkosten erweiterten Kosten.

Skalierung: $c_{\max} = \sum_{j=1}^n c_j = 51$ Summe der Kosten aller Spalten

Kosten einer Lösung x^i , $i = 1, \dots, p$: $C(x^i)$ (inklusive Strafkosten)

$$f^i := f(x^i) = \begin{cases} c_{\max} - c(x^i), & \text{falls } c(x^i) < c_{\max} \\ 0 & , \text{sonst} \end{cases}$$

$$f^i \geq 0$$

Selektion mit Roulette – Wheel

(Individuen die Wahrscheinlichkeit 0
selektiert werden sind ggf. dabei)

2.5. Genetische und Evolutionäre Algorithmen (GEA)

Binärkodierung, einfacher Crossoveroperator und Mutation, Roulette

Ausgangsposition mit $p = 4$ sei zufällig erzeugt

Name	Chromosom	Kosten	Fitness	
x^1	(0 0 1 0 1 0 0 1 1)	28	23	
x^2	(1 0 1 0 1 1 0 0 0)	24	27	
x^3	(1 0 0 0 0 0 1 0 0)	48	3	Fitnesssumme = 86
x^4	(1 1 0 1 0 0 0 0 1)	18	33	

↖ Strafkosten von $2 \cdot 20$ eingeschlossen, x^3 ist nicht zulässig für das Set-Covering Problem

Name	Selektionswahrscheinlichkeit	Gewählte Anzahl	
x^1	$23/86 \approx 0,267$	1	Die Individuen seien in der Reihenfolge x^1, x^4, x^2, x^3 selektiert worden!
x^2	$27/86 \approx 0,314$	1	
x^3	$3/86 \approx 0,035$	0	
x^4	$33/86 \approx 0,384$	2	

2.5. Genetische und Evolutionäre Algorithmen (GEA)

Deshalb werden die Paare (x^1, x^4) und (x^2, x^4) gebildet und danach der [Crossover-Operator](#) angewendet:

Rolle	Name	Kreuzungsposition	Chromosom
Elternteil	x^1	3	(0 0 1 / 0 1 0 0 1 1)
Elternteil	x^4		(1 1 0 / 1 0 0 0 0 1)
<hr style="border-top: 1px dashed black;"/>			
Kind	x_n^1		(0 0 1 / 1 0 0 0 0 1)
Kind	x_n^2		(1 1 0 / 0 1 0 0 1 1)
<hr style="border-top: 3px double black;"/>			
Elternteil	x^2	7	(0 1 0 0 1 1 0 / 0 0)
Elternteil	x^4		(1 1 0 1 0 0 0 / 0 1)
<hr style="border-top: 1px dashed black;"/>			
Kind	x_n^3		(1 0 0 0 1 1 0 / 0 1)
Kind	x_n^4		(1 1 0 1 0 0 0 / 0 0)

2.5. Genetische und Evolutionäre Algorithmen (GEA)

Mutation

Ergebnis sei, dass nur ein Kind, Kind x_n^3 an Position 5 mutiert wird (geringe Mutationswahrscheinlichkeit): $x^3 = (1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1)$

Abschluss der Iteration ($x^1 := x_n^1$, $x^2 := x_n^2$, $x^3 = (1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1)$, $x^4 := x_n^4$)

2. Iteration

Berechnung der Fitness der Individuen

Name	Chromosom	Kosten	Fitness
x^1	(0 0 1 1 0 0 0 0 1)	21	30
x^2	(1 1 0 0 1 0 0 1 1)	25	26
x^3	(1 0 1 0 1 1 0 0 1)	21	30
x^4	(1 1 0 1 0 0 0 0 0)	14	37

(Deutlich erhöhte Fitness gegenüber Ausgangspopulation, gilt natürlich nicht generell!)

2.5. Genetische und Evolutionäre Algorithmen (GEA)

Verschiedene Aspekte der GEA

- Hauptschwierigkeit, Zulässigkeit der “Kinderlösungen“ zu gewährleisten
 - Zulässigkeit durch genetische Operatoren direkt garantieren
(→ kompliziertere Operatoren)
 - Unzulässige Lösungen einbeziehen und mit Strafkosten belegen
(→ wie Strafkosten wählen?)
 - Jedes “Kind“ auf Zulässigkeit untersuchen. Wenn unzulässig
Reparaturalgorithmus zur Herstellung der Zulässigkeit anwenden.
(→ gelingt “Reparatur“ nicht, wird Lösung verworfen)
- Mathematische Analyse der GEA
“Schema – Theorem“ von Holland! (weiterführende Literatur)
- Anordnung der Gene im Chromosom
z. B. mit “Umordnungsgeneratoren“ eine gegebene Reihenfolge der Gene im Chromosom zufällig verändern, um schlecht gewählte Anfangsreihenfolgen im Evolutionsprozess zu verändern.

3.1. Netzflußprobleme

Optimierungsprobleme auf Netzwerken

(hier: Betrachtung auf gerichteten Graphen)

Auswahl von Problemen mit besonderer Relevanz für die Transportlogistik

Das Minimalkosten – Netzflußproblem

~ Grundmodell der Netzflußtheorie (engl. minimum cost flow, MCF) (wird auch als “Umladeproblem“ transshipment problem bezeichnet)

Viele bekannte Modelle ergeben sich durch Spezialisierung aus dem MCF.

3.1. Netzflußprobleme

Gegeben sei ein Netzwerk $D = (V, A, c, l, u, b)$

V – Knotenmenge, A – Menge der (gerichteten) Bögen, $|A| = n$

- Zu jedem $(i, j) \in A$ sind Kosten c_{ij} definiert: Variable Kosten für den Transport einer Flußeinheit auf dem Bogen (i, j)

Die Kosten seien als linear von der Flußmenge abhängig angenommen.

(Verallgemeinerung z.B. auf konvexe bzw. konkave Kostenfunktionen seien möglich.)

- Jeder Bogen $(i, j) \in A$ besitzt eine Kapazität $u_{ij} > 0$, die angibt, wieviele Flußeinheiten maximal auf (i, j) transportiert werden können.
- Untere Schranken l_{ij} für den Fluß in $(i, j) \in A$ können ebenfalls definiert sein:

$$0 \leq l_{ij} \leq u_{ij}$$

3.1. Netzflußprobleme

- Zu jedem Knoten $i \in V$ gibt eine Größe b_i ein Angebot oder eine Nachfrage nach den (dem) fließenden Gütern (Gut) an.

$b_i > 0$: Knoten i heißt Angebotsknoten (Fabriken, Fahrzeugdepots, Frachtzentren)
(Einspeisen des Gutes in das Netzwerk)

$b_i < 0$: Knoten i heißt Nachfrageknoten (Kunden, Frachtempfänger,
Stromabnehmer)

$b_i = 0$: Umladeknoten i (transshipment nodes) (Lager, Hubs, Fahrzeugtauschpunkte)

Optimierungsmodell

x_{ij} – Flußvariablen ~ Fluß zu Bogen $(i, j) \in A$

$\delta^+(i) = \{(j, i) \mid j \in V \text{ und } (j, i) \in A\}$ in i einlaufende Bögen

$\delta^-(i) = \{(i, j) \mid j \in V \text{ und } (i, j) \in A\}$ aus i ausgehende Bögen

3.1. Netzflußprobleme

Modell MCF

$$\min z_{\text{MCF}} = \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (1)$$

$$\text{so dass:} \quad \sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = b_i \quad (2)$$

für alle $i \in V$ Flußbilanzgleichungen

$$(0 \leq) l_{ij} \leq x_{ij} \leq u_{ij} \quad \text{für alle } (i, j) \in A \quad (3)$$

Kapazitätsrestriktionen für die Flüsse

Das Modell hat nur dann eine zulässige Lösung wenn gilt:

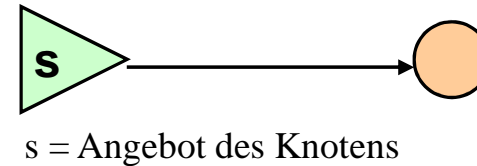
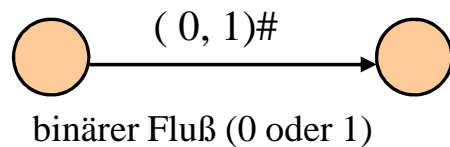
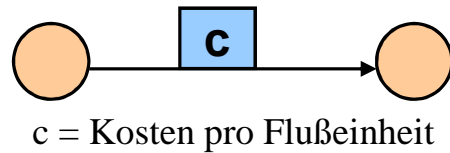
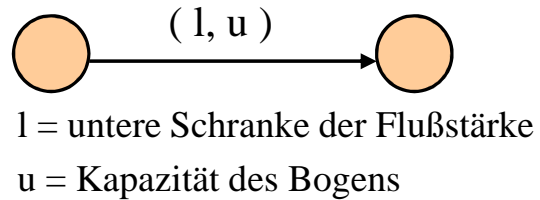
$$\text{Angebot} = \text{Nachfrage} \approx \sum_{i \in V} b_i = 0 \quad (4)$$

3.1. Netzflußprobleme

Eigenschaften des MCF – Problems (Modell (1) – (3))

- Die Lösungen des MCF – Problems sind ganzzahlig, wenn alle Daten ganzzahlig sind (alle l_{ij} , u_{ij} und b_i)
(Grund: Unimodularität der Inzidenzmatrix)
- MCF – Probleme können mit effizienten Algorithmen gelöst werden. Da MCF-Probleme LoP's sind können sie insbesondere mit dem Simplexalgorithmus bzw. mit speziellen Versionen des Simplexalgorithmus gelöst werden.
- Graphische Darstellung in der Ebene unter Verwendung der nachfolgenden dargestellten Symbole

3.1. Netzflußprobleme



Spezialisierung des MCF Problems erzeugt eine Reihe bekannter Standardprobleme des OR:

- das Kürzeste-Wege Problem
- das Maximalflußproblem
- das Zuordnungsproblem \Rightarrow (QM(OR), QM der Wiwi)
- das Transportproblem \Rightarrow (siehe 3.2)

3.1. Netzflußprobleme

Das Kürzeste – Wege – Problem (Shortest Path Problem, SPP)

Von einem ausgezeichneten Knoten s (Quelle) sucht man zu einem weiteren spezifizierten Knoten t (Senke) einen kürzesten Weg

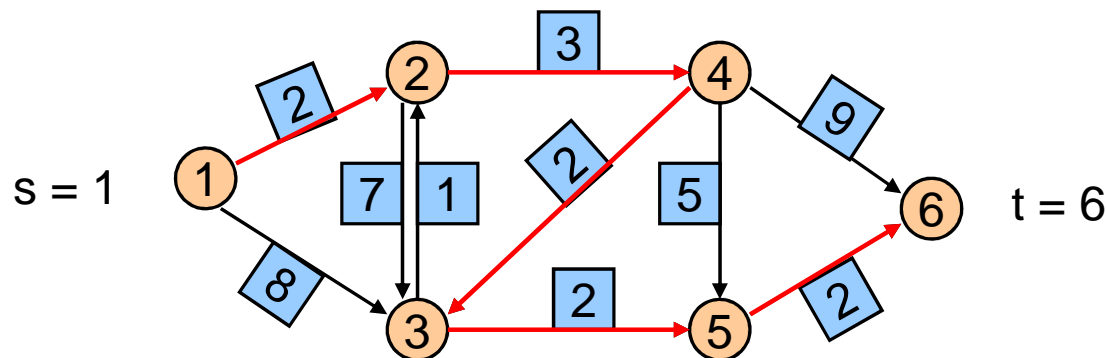
“Kurz“ - bezieht sich auf die “Kostenbewertung“ der Bögen.

Das SPP ist ein MCFP mit:

$l_{ij} = 0, u_{ij} = 1$ für alle $(i, j) \in A$

$b_s = +1$ (Angebot der Quelle), $b_t = -1$ (Nachfrage der Senke)

Alle anderen Knoten i sind Umschlagknoten, $b_i = 0$.

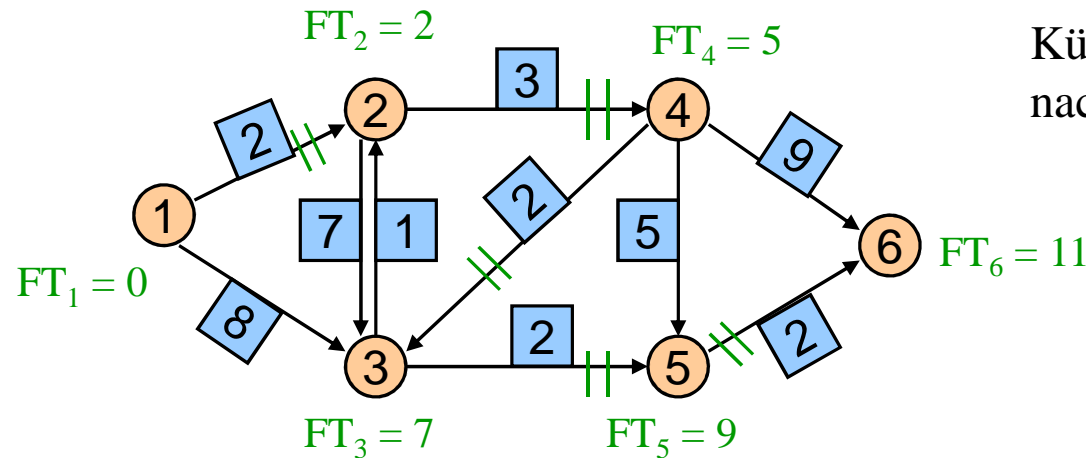


Kürzester Weg von Quelle s zu
Senke t in bewerteten
Digraphen:

→: $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 5 \rightarrow 6$

Länge: 11

3.1. Netzflußprobleme



Kürzester Weg von Quelle 1
nach Senke 6

SPP sind einfach zu lösen solange keine Zyklen mit negativer Bewertung auftreten (sonst würde der Weg immer kürzer je öfter man den Zyklus durchläuft)

Verfahren:

- Label – Setting – Algorithmen (z.B. Dijkstra – Algorithmus)
- Label – Correcting – Algorithmen (z.B. FIFO – Algorithmus)
- Dynamische Optimierung für SPP mit Zeitfenstern

(Grünert und Irnich: Optimierung im Transport, Band II, Kapitel 4, Shaker Verlag, 2005)

3.1. Netzflußprobleme

Mehrgüterflüsse

Bisher: Nur ein Gut betrachtet, welches durch die Bögen des Netzwerkes fließt.

Jetzt: Menge verschiedener Güter fließt durch das Netzwerk (z.B. Briefe, Pakete, Fracht)

~ Mehrgüterflüsse nutzen (konkurrieren um) eine gemeinsame Ressource c die durch die Bogen-Kapazität beschränkt ist. (sonst könnte man jedes Gut einzeln betrachten)

MCNF – Multi commodity network flow problems

Modell: Gerichteter Graph $G = (V, A)$

- Durch die Bögen des Digraphen können Güter $k = 1, 2, \dots, K$ fließen
- Kosten für den Transport einer Flusseinheit des Gutes k durch den Bogen (i, j) seien mit c_{ij}^k gegeben.
- Jeder Bogen hat eine güterspezifische Kapazität $u_{ij}^k \geq 0$

3.1. Netzflußprobleme

- Untere Schranken l_{ij}^k sind ebenfalls möglich.
=> Gemeinsame Kapazitäten (zusätzlich zu den güterspezifischen Kapazitäten) für den Gesamtfluß in einem Bogen.
$$0 \leq L_{ij} \leq U_{ij}$$
- Zu jedem Knoten $i \in V$ und jedem Gut $k \in \{1, \dots, K\}$ gibt es eine reellwertige Größe b_i^k (Angebot bzw. Nachfrage nach Gut k im Knoten i)
- Entscheidungsvariablen x_{ij}^k : ~ Flußvariablen d.h. Fluß des Gutes k durch Bogen i

3.1. Netzflußprobleme

Modell: MCNF

$$\min z_{\text{MCNF}} = \sum_{k=1}^K \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k \quad (5)$$

$$\text{so dass:} \quad \sum_{(i,j) \in \delta^+(i)} x_{ij}^k - \sum_{(j,i) \in \delta^-(i)} x_{ji}^k = b_i^k \quad (6)$$

für alle $i \in V, k \in \{1, \dots, K\}$

$$l_{ij}^k \leq x_{ij}^k \leq u_{ij}^k \quad \text{für alle } (i,j) \in A, k \in \{1, \dots, K\} \quad (7)$$

$$(*) \quad L_{ij} \leq \sum_{k=1}^K x_{ij}^k \leq U_{ij} \quad \text{für alle } (i,j) \in A \quad (8)$$

Das Modell besitzt nur dann eine zulässige Lösung, wenn $\sum_{i \in V} b_i^k = 0$ (für alle k) gilt.

(*) neu: gemeinsame Kapazitätsrestriktionen

3.2 Das Transportproblem

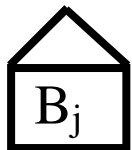
3.2.1 Modellierung

Voraussetzung: Ein (homogenes) Gut: „Erzeugnis“

Problem:



1. m Ausgangsorte A_i , $i = 1, \dots, m$ (Lager, Produzenten),
in denen ein Erzeugnis zur Verfügung steht
 a_i - Mengeneinheiten (Aufkommen)



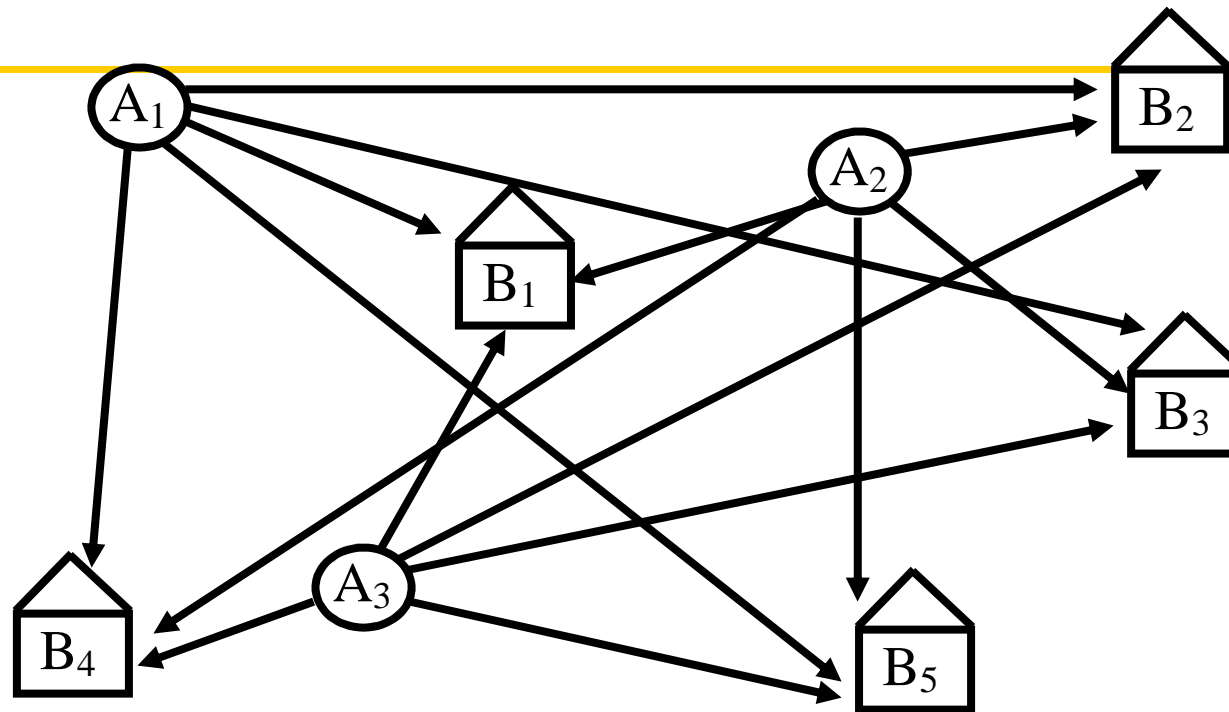
2. n Bestimmungsorte B_j , $j = 1, \dots, n$ (Kunden, Produzenten, Lager),
die einen Bedarf an diesem Erzeugnis haben
 b_j - Mengeneinheiten (Bedarf)

Belieferung direkt von jeweils einem A_i zu einem B_j .

Relation:



c_{ij} : Transportkosten pro Mengeneinheit ($m \cdot n$ Relationen)



Voraussetzung: o.E.d.A.

Der Gesamtbedarf ist gleich dem Gesamtaufkommen: $\sum_{i=1}^m a_i = \sum_{j=1}^n b_j$

Problemformulierung (verbal):

Das Aufkommen an jedem Ausgangsort ist so zu den Bedarfsorten zu transportieren, dass der Bedarf jeweils komplett gedeckt wird und der Transportplan minimale Kosten besitzt.

3.2. Das Transportproblem

Modell

Transportierte Menge des Erzeugnisses von A_i nach B_j sei x_{ij} ,
 $i = 1, 2, \dots, m$; $j = 1, 2, \dots, n$ (in Mengeneinheiten des Erzeugnisses).

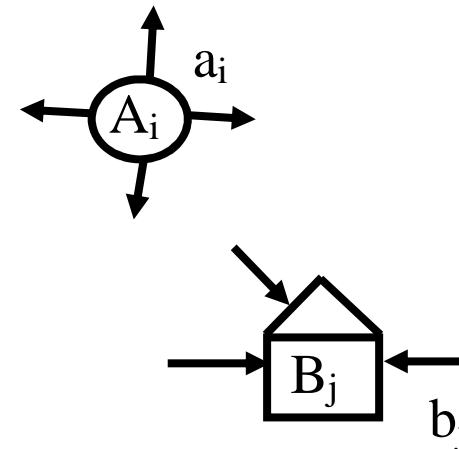
$$\text{minimiere } z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} \cdot x_{ij}$$

bezüglich der Restriktionen

$$\sum_{j=1}^n x_{ij} = a_i \text{ für } i = 1, 2, \dots, m$$

$$\sum_{i=1}^m x_{ij} = b_j \text{ für } j = 1, 2, \dots, n$$

$$x_{ij} \geq 0, \text{ für } i = 1, 2, \dots, m \text{ und } j = 1, 2, \dots, n$$



3.2. Das Transportproblem

Modell (*)

wobei gilt: $\sum_{i=1}^m a_i = \sum_{j=1}^n b_j$

Ausgeglichenheit des Problems

3.2.2 Beispiel und Eröffnungsheuristik

	B ₁	B ₂	B ₃	B ₄
A ₁	1	8	4	7
A ₂	9	0	5	7
A ₃	3	6	8	1

Kostenmatrix $C = ((c_{ij}))$
 $i = 1, \dots, m; j = 1, \dots, n$

3.2. Das Transportproblem

	B ₁	B ₂	B ₃	B ₄	a _i
A ₁	x ₁₁	x ₁₂	x ₁₃	x ₁₄	20
A ₂	x ₂₁	x ₂₂	x ₂₃	x ₂₄	25
A ₃	x ₃₁	x ₃₂	x ₃₃	x ₃₄	40
b _j	10	25	15	35	85

Matrix der Variablen +
Aufkommen / Bedarf

	B ₁	B ₂	B ₃	B ₄
A ₁	1	8	4	7
A ₂	9	0	5	7
A ₃	3	6	8	1

Kostenmatrix $C=((c_{ij}))$
 $i=1,...,m; j=1,...,n$

3.2. Das Transportproblem

1. Greedy (minimales Element maximal belegen!)

	B ₁	B ₂	B ₃	B ₄	a _i
A ₁	10	-	10	-	20
A ₂	-	25	-	-	25
A ₃	-	-	5	35	40
b _j	10	25	15	35	85

$m + n = 7$ ($m + n - 1$ BV), hier 5 Variablen > 0

Zielfunktion:

$$10 \cdot 1 + 10 \cdot 4 + 25 \cdot 0 + 5 \cdot 8 + 35 \cdot 1 = 10 + 40 + 40 + 35 = 125$$

3.2. Das Transportproblem

2. NordWestEcken-Regel

- a) Beginne in der NWE und setze $x_{11} = \min(a_1, b_1)$
- b) falls $x_{11} = a_1$, setze $x_{21} = \min(a_2, b_1 - x_{11})$
 falls $x_{11} = b_1$, setze $x_{12} = \min(a_1 - x_{11}, b_2)$
 Ist $x_{11} = a_1 = b_1$, setze zusätzlich $x_{12} = 0$ oder $x_{21} = 0$ als Basiseintrag („Basis-Null“)
- c) Iteriere bis alle x_{mn} festgelegt sind. Die entstehende Lösung ist eine ZBL für das Transportmodell.

	B ₁	B ₂	B ₃	B ₄	a _i
A ₁	10	10	-	-	20
A ₂	-	15	10	-	25
A ₃	-	-	5	35	40
b _j	10	25	15	35	85

3.2. Das Transportproblem

a) $x_{11} = \min(a_1, b_1) = \min(20, 10) = 10 = b_1$

b) $x_{12} = \min(a_1 - x_{11}, b_2) = \min(10, 25) = 10$ usw...

$$ZF = 10 \cdot 1 + 10 \cdot 8 + 10 \cdot 5 + 5 \cdot 8 + 35 \cdot 1 = 10 + 80 + 50 + 40 + 35 = 215$$

3. VAM-Heuristik

I : Indexmenge der noch nicht erfüllten Zeilenbedingungen

J : Indexmenge der noch nicht erfüllten Spaltenbedingungen

- a) Bestimme für alle noch nicht erfüllten Zeilen und Spalten die Opportunitätskosten. Sie sind gleich der Differenz (D_i bzw. \bar{D}_j) zwischen den zweitniedrigsten und den niedrigsten Einheitstransportkosten c_{ij} ($i \in I, j \in J$).
- b) Bestimme in der Zeile oder Spalte mit der maximalen Differenz das Feld (i, j) mit dem kleinsten c_{ij} . Falls die maximale Differenz nicht eindeutig ist, so wähle eine dieser Zeilen oder Spalten beliebig aus.
-

3.2. Das Transportproblem

- c) Im Feld (i, j) setze $x_{ij} = \min(a_i, b_j)$, so dass eine Spalten- oder Zeilenbedingung erfüllt ist. Streiche die erfüllte Zeile oder Spalte und reduziere die nicht erfüllte Forderung (a_i oder b_j) um den Wert x_{ij} . Falls beide Bedingungen erfüllt sind, streiche entweder die Zeile oder die Spalte (in der übrig gebliebenen Zeile oder Spalte entsteht eine $BV=0$).
- d) Überprüfe, ob alle Zeilen- oder Spaltennebenbedingungen erfüllt sind. Wenn ja "STOP", sonst "gehe zu a)".

Beispiel:

	B_1	B_2	B_3	B_4
A_1	1	8	4	7
A_2	9	0	5	7
A_3	3	6	8	1

- a) Bestimmung der Opportunitätskosten D_i, \bar{D}_j für alle Zeilen $i \in I$ und alle Spalten $j \in J$.

	B_1	B_2	B_3	B_4	a_i	D_i
A_1	1	8	4	7	20	3
A_2	9	0	5	7	25	5
A_3	3	6	8	1	40	2
b_j	10	25	15	35	85	
\bar{D}_j	2	6	1	6		

- b) Höchste Opportunitätskosten sind $\bar{D}_j = \bar{D}_2 = \bar{D}_4 = 6$
nicht eindeutig: Spalte 2 wird gewählt, Feld (2, 2) führt zu minimalen $c_{ij} = 0 = c_{22}$

c) $x_{22} = \min(25, 25) = 25$

Beide Bedingungen sind erfüllt. Wir streichen Spalte, Index $j=2$ aus J und setzen $a_2^{\text{neu}} = a_2^{\text{alt}} - x_{22} = 0 \rightarrow$ eine Schleife der Iteration beendet, da Schritt d) mit "sonst" ausgeht
Zurück zu a)

3.2. Das Transportproblem

zweiter Durchlauf:

$$I = \{1, 2, 3\}, J = \{1, 3, 4\}$$

	B ₁	B ₂	B ₃	B ₄	a _i	D _i ⁽²⁾
A ₁	1	-	4	7	20	3
A ₂	9	(25) ¹	(0) ^{1'} ₅	7	0	2
A ₃	3	-	8	1	40	2
b _j	10	0	15	35	60	
$\bar{D}_j^{(2)}$	2	-	1	6		

$\bar{D}_j^{(2)} = 6$ größte Opportunitätskosten, Feld (3,4) kleinstes c_{ij} (= 1)

Zuweisung: $x_{34} = 35$, $a_{\text{neu}}^3 = 40 - 35 = 5$

Index $j = 4$ aus J streichen.

3.2. Das Transportproblem

dritter Durchlauf:

$$I = \{1, 2, 3\}, J = \{1, 3\}$$

	B ₁	B ₂	B ₃	B ₄	a _i	D _i ⁽³⁾
A ₁	1	-	4	-	20	3
A ₂	9	(25) ¹	(0) ^{1'} ₅	-	0	4
A ₃	3	-	8	(35) ²	5	5
b _j	10	0	15	0	25	
$\bar{D}_j^{(3)}$	2	-	1	-		

$$\max_{i,j} (D_i^{(3)}, \bar{D}_j^{(3)}) = 5 = D_3^{(3)}$$

$c_{31} = 3$ kleinstes c_{ij}

$x_{31} = 5$; Reduktion: $b_1^{(\text{neu})} = 10 - 5$; $I = \{1, 2\}$ usw.

5. Iteration liefert optimale Lösung

3.2.3 Die Modi-Methode

$$\sum_{i=1}^m a_i = \sum_{j=1}^n b_j \quad \text{ausgeglichenes Problem und } a_i, b_j \geq 0$$

Darstellung als LP

		x_{11}	x_{12}	...	x_{1n}	x_{21}	x_{22}	...	x_{2n}	...	x_{m1}	x_{m2}	...	x_{mn}	RS
Zeile	1	1	1	...	1	0	0	...	0	...	0	0	...	0	a_1
	2	0	0	...	0	1	1	...	1	...	0	0	...	0	a_2
	...														
	m	0	0	...	0	0	0	...	0	...	1	1	...	1	a_m
Zeile	1	1	0	...	0	1	0	...	0	...	1	0	...	0	b_1
	2	0	1	...	0	0	1	...	0	...	0	1	...	0	b_2
	...														
	n	0	0	...	1	0	0	...	1	...	0	0	...	1	b_n

3.2.3 Die Modi-Methode

$$\sum_{j=1}^n x_{ij} = a_i \quad \text{für } i = 1, 2, \dots, m \quad (1)$$

$$\sum_{i=1}^m x_{ij} = b_j \quad \text{für } j = 1, 2, \dots, n \quad (2)$$

m·n Spalten und m+n Zeilen

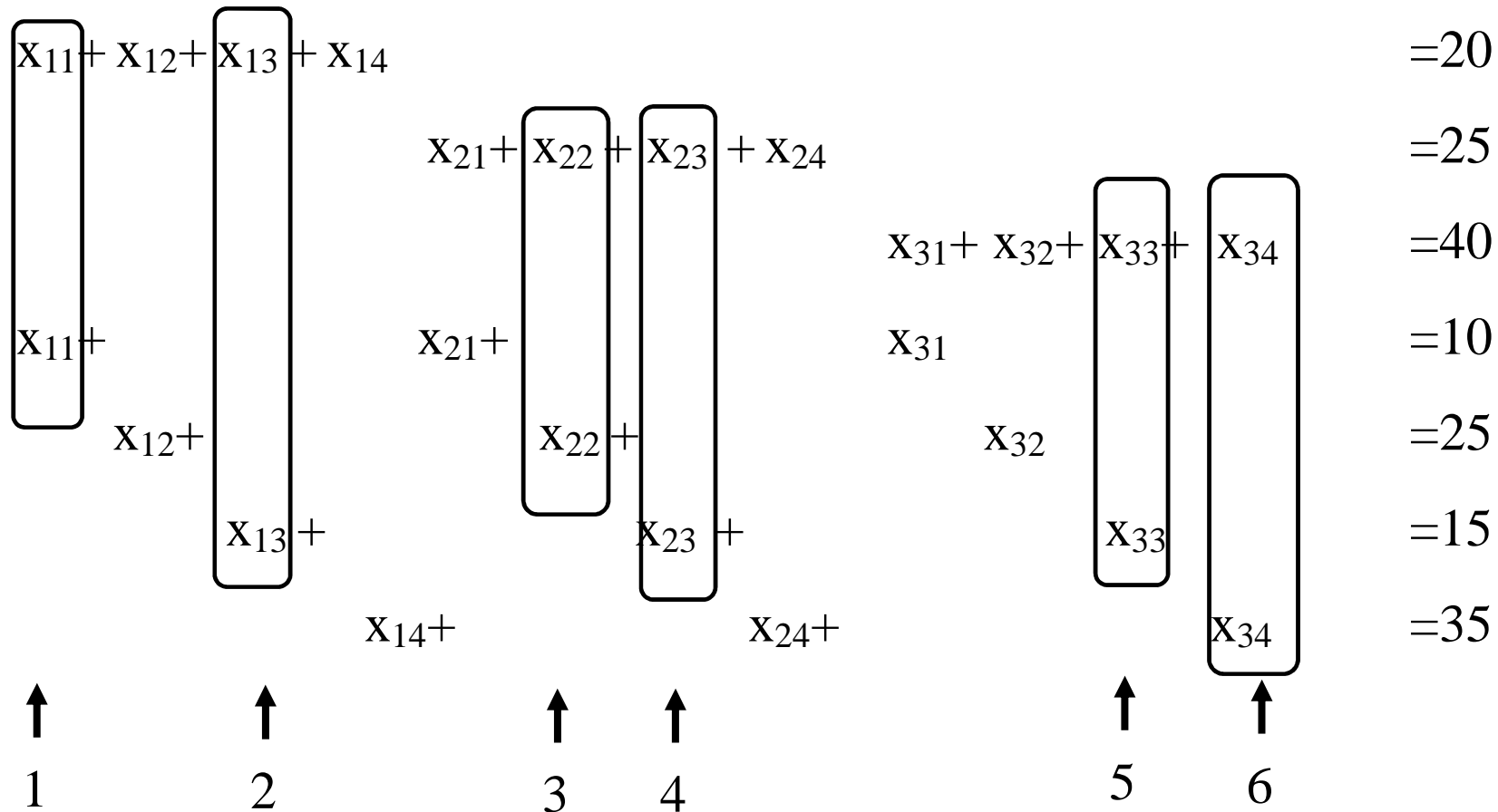
Lösung mit der Simplexmethode:

Tableau hätte m+n Zeilen, m·n Spalten;

Basis (m+n-1)·(m+n-1) Teilmatrix vom Rang m+n-1.

Wegen Ausgeglichenheit des Transportproblems genügen m+n-1 BV.

3.2.3 Die Modi-Methode



3.2.3 Die Modi-Methode

$$\begin{array}{rclcl}
 x_{11} + & x_{13} & & & = 20 \\
 & & x_{22} + & x_{23} & = 25 \\
 & & & & x_{33} + x_{34} = 40 & (\text{Alle NBV}=0) \\
 x_{11} & & & & = 10 \\
 & & x_{22} + & & = 25 \\
 & x_{13} + & & x_{23} + & x_{33} & = 15 \\
 & & & & & x_{34} = 35
 \end{array}$$

Lösung:

$$x_{11} = 10, x_{34} = 35, x_{13} = 10, x_{33} = 5, x_{22} = 25, x_{23} = 0$$

6 BV sind wegen der Ausgeglichenheit ausreichend.

3.2.3 Die Modi-Methode

Spezielle Struktur der Koeffizientenmatrix A (nur Nullen und Einsen, speziell angeordnet) erlaubt folgende Schlußfolgerungen:

1. B^{-1} enthält ausschließlich $+1, -1, 0$
2. $B^{-1}b = x_B$ ganzzahlig, falls b ganzzahlig
3. Ein vereinfachtes Tableau ist möglich, da die Strukturen von A, B, B^{-1} bekannt sind.
4. Es genügt, in jeder Iteration die Basis, die Basislösung und die Δz_j der NBV zu bestimmen!

3.2.3 Die Modi-Methode

Transporttableau

	B_1	B_2	...	B_n	a_i
A_1	x_{11}	x_{12}	...	x_{1n}	a_1
A_2	x_{21}	x_{22}	...	x_{2n}	a_2
...
A_m	x_{m1}	x_{m2}	...	x_{mn}	a_m
b_j	b_1	b_2	...	b_n	$\sum a_i = \sum b_j$

Zulässige
Ausgangsbasislösung:
z.B. mittels NWR

Verbesserungsverfahren:

MODI (Modified Distribution Method), Vajda 1962 (auch U-V Methode)

Neumann/Morlock 330ff, Zimmermann 128ff

Grundidee: Verwendung des Dualen Modells zu (*)

$$(**) \text{ maximiere } Z = \sum_{i=1}^m a_i u_i + \sum_{j=1}^n b_j v_j \quad u_i + v_j \leq c_{ij}, \text{ für } i=1, \dots, m; j=1, \dots, n$$

3.2.3 Die Modi-Methode

Die dualen Strukturvariablen sind wegen der Gleichungsrestriktionen des PP (Modell *) unbeschränkt.

Gleichungssystem des PP ist unterbestimmt → einer dualen Variablen kann ein beliebiger Wert zugeordnet werden.

Wegen der Complementary Slackness Bedingung gilt für die BV: $u_i + v_j = c_{ij}$.

Da die Basis $m+n-1$ BV enthält, ist das Gleichungssystem $u_i + v_j = c_{ij}$ unterbestimmt. Man legt $u_1=0$ (oder $v_1=0$) fest und löst danach das Gleichungssystem für alle BV.

Für die aufzunehmende NBV benötigt man Δz_{ij} . Diese sind hier gleich $\Delta z_{ij} = c_{ij} - (u_i + v_j)$.

Optimalitätsbedingung: $\Delta z_{ij} \geq 0$ für alle NBV

Aufnahmeregel: Diejenige NBV x_{rs} in die Basis aufnehmen, für die $\Delta z_{rs} = \min_{i,j} \Delta z_{ij} < 0$ gilt.

3.2.3 Die Modi-Methode

Basistausch - zu eliminierende BV:

Im Transporttableau Pfad aus denjenigen BV bestimmen, deren Wert sich bei Erhöhung des Wertes der aufzunehmenden NBV ändert, damit die Restriktionen erhalten bleiben (~Stepping-Stone-Path). Dieser Pfad ist eindeutig. Die entsprechenden BV seien x_{ij}^B . Durch Hinzufügen der NBV x_{rs} entsteht eine Schleife. Dann gilt:

1. Die Werte der Variablen x_{ij}^B im SSP ändern sich entweder um $+x_{rs}$ oder $-x_{rs}$.
2. Das Vorzeichen alterniert (Markierung mit + bzw. - genügt). Sobald bei Erhöhung der NBV eine mit - (minus) markierte ursprüngliche BV einen Wert <0 annimmt, würde Unzulässigkeit eintreten. Deshalb:

Ausscheidende BV: x_{kl}^B gemäß $\theta = \min_{i,j} x_{ij}^B = x_{kl}^B$ bestimmen.
(- markiert)

3.2.3 Die Modi-Methode

θ gibt auch den Wert der aufzunehmenden NBV an (Pivotisierungsprozess).

Beispiel: NordWestEcken-Regel

→ Ausgangsbasislösung

	B ₁	B ₂	B ₃	B ₄	a _i
A ₁	10	10			20
A ₂		15	10		25
A ₃			5	35	40
b _j	10	25	15	35	85

NBV=0; leere Felder

$u_i + v_j = c_{ij}$ mit $v_1 = 0$ für alle BV lösen

3.2.3 Die Modi-Methode

$$u_1 + v_1 = c_{11} = 1 \quad \text{wegen } v_1 = 0 \quad \rightarrow u_1 = 1$$

$$u_1 + v_2 = c_{12} = 8 \quad \rightarrow v_2 = 7$$

$$u_2 + v_2 = c_{22} = 0 \quad \rightarrow u_2 = -7$$

$$u_2 + v_3 = c_{23} = 5 \quad \rightarrow v_3 = 12$$

$$u_3 + v_3 = c_{33} = 8 \quad \rightarrow u_3 = -4$$

$$u_3 + v_4 = c_{34} = 1 \quad \rightarrow v_4 = 5$$

Berechnung der $\Delta z_{ij} = c_{ij} - (u_i + v_j)$ für alle NBV:

Schema:

- In Klammern c_{ij}
- Alle BV entsprechenden Felder mit $\Delta z_{ij} = 0$ belegen
- u_i und v_j Spalte bzw. Zeile anfügen
→ Δz_{ij} für NBV berechnen.

3.2.3 Die Modi-Methode

	B ₁	B ₂	B ₃	B ₄	u _i
A ₁	(1) 0	(8) 0	(4) -9	(7) 1	1
A ₂	(9) 16	(0) 0	(5) 0	(7) 9	-7
A ₃	(3) 7	(6) 3	(8) 0	(1) 0	-4
v _j	0	7	12	5	

Wegen $\Delta z_{13} = -9 < 0$ ist die BL nicht optimal.

x_{13} in die Basis aufnehmen! Die x_{13} als NBV enthaltende Schleife zur BL ist.

	B ₁	B ₂	B ₃	B ₄	a _i
A ₁	10	10- ← +			20
A ₂		↓ 15+ → 10-			25
A ₃			5	35	40
b _j	10	25	15	35	85

3.2.3 Die Modi-Methode

$$\theta = \min x_{ij}^B = 10$$

Schleife entsteht für $x_{12} = x_{23} = 10$.

Pivotisierung: Ändern der Elemente der Schleife im Sinne der Markierung um 10.

	B ₁	B ₂	B ₃	B ₄	a _i
A ₁	10		10		20
A ₂		25	0		25
A ₃			5	35	40
b _j	10	25	15	35	85

Entartete BL, da weniger als $m+n-1=6$ BV >0 sind. Frei entscheiden, ob x_{12} oder x_{23} als BV (Basisnull) behandelt wird.
(Ist übrigens Greedy Lösung!)

3.2.3 Die Modi-Methode

Optimale Lösung nach einer weiteren Iteration:

$$x_{11} = 5, \quad x_{13} = 15, \quad x_{22} = 25, \quad x_{23} = 0, \quad x_{31} = 5, \quad x_{34} = 35$$

$$\text{mit } Z = 1 \cdot 5 + 4 \cdot 15 + 3 \cdot 5 + 1 \cdot 35 = 5 + 60 + 15 + 35 = 115$$

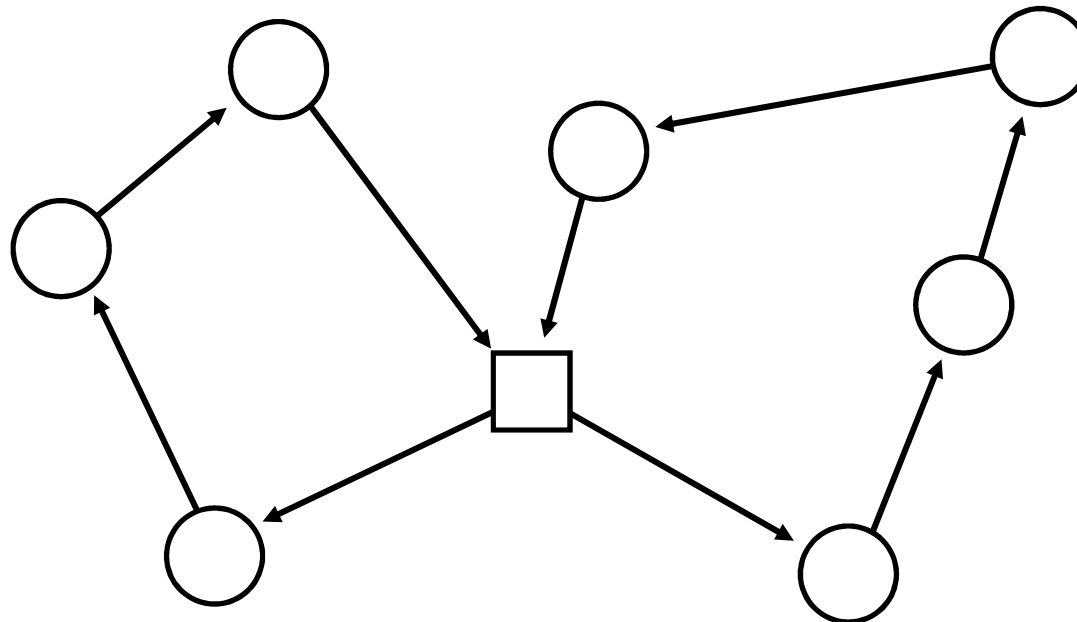
(Diese Lösung wird bei diesem Beispiel auch durch die VAM geliefert.)

3.3. VRP: Modellierung und heuristische Lösung

3.3. Das Vehicle-Routing-Problem (VRP): Modellierung und heuristische Lösung

3.3.1. Modelle zum VRP

Definition des Vehicle-Routing-Problems



3.3. VRP: Modellierung und heuristische Lösung

Graph $G = (V, A, C)$

- Knotenmenge $V = \{1, \dots, n\}$:
 - Depots $R = \{1, \dots, r\}, r < n$
 - Kunden $i \in V \setminus R = \{r+1, \dots, n\}$ mit Nachfrage d_i
- Kantenmenge A : gerichtete oder ungerichtete Kanten
- Kostenmatrix $C = (c_{ij})$
 - C symmetrisch: $c_{ij} = c_{ji}, \forall i, j \in V$ bzw. C asymmetrisch
 - C euklidisch: $c_{ij} + c_{jk} \geq c_{ik}, \forall i, j, k \in V$

Nebenbedingungen für das Vehicle-Routing-Problem

- Kapazitätsbeschränkung der Fahrzeuge: D oder $D_k, k = 1, \dots, m$
Nachfrage in den Knoten: $d_i \geq 0, \forall i \in V, d_i = 0 \forall i \in R$

$$\sum_{i \in V \setminus R} d_i \leq D \cdot m \wedge d_i \leq D \forall i \in V$$

3.3. VRP: Modellierung und heuristische Lösung

- Anzahl der Orte auf einer Route
- Zeitbeschränkungen: Reisezeit und Wartezeit
- Zeitfenster: Ort i muß im Intervall $[a_i, b_i]$ angefahren werden
- Reihenfolge der zu bedienenden Orte: i vor j

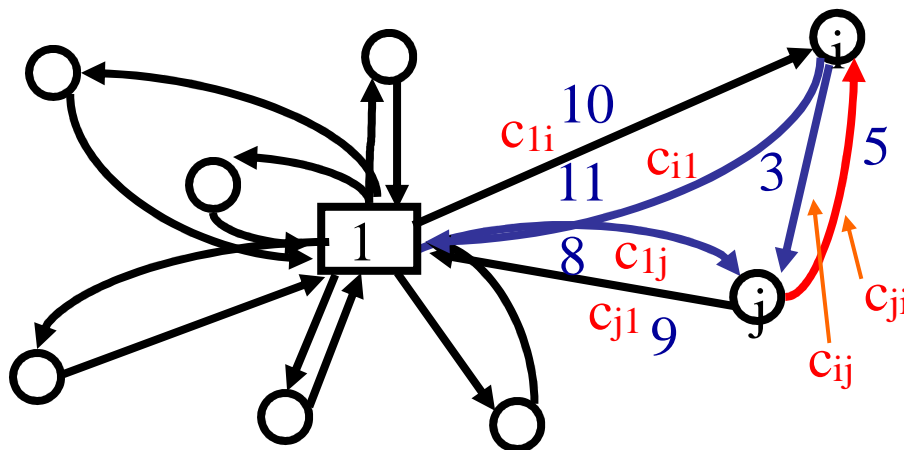
Modelleigenschaften des Vehicle-Routing-Problems

- Fahrzeugflotte: m Fahrzeuge in den Depots R
 - m fest vorgegeben oder variabel: $\underline{m} \leq m \leq \bar{m}$
 - Anzahl der Fahrzeuge pro Depot
 - Fixkosten für die Benutzung eines Depots oder Fahrzeugs
- Routenaufbau: Kostenminimale Fahrzeugrouten
 - Besuche jeden Ort aus $V \setminus R$ genau einmal mit einem Fahrzeug
 - Jedes Fahrzeug kehrt an seinen Ausgangspunkt zurück
 - Weitere Nebenbedingungen

3.3. VRP: Modellierung und heuristische Lösung

3.3.2. Eröffnungsheuristiken (Savings- und Sweep-Verfahren)

1. Savings-Methode (Methode von Clarke and Wright)



Symmetrischer Fall:

$$c_{1i} = c_{i1}, c_{1j} = c_{j1}, c_{ij} = c_{ji}$$

Savings: s_I, s_{II}

Kosten-Pendeltouren: $c_{1i} + c_{i1} + c_{1j} + c_{j1}$

Kosten-Verkettete Tour:I: $c_{1i} + c_{ij} + c_{j1}$

II: $c_{1j} + c_{ji} + c_{i1}$

$$s_I = c_{i1} + c_{1j} - c_{ij} = 16$$

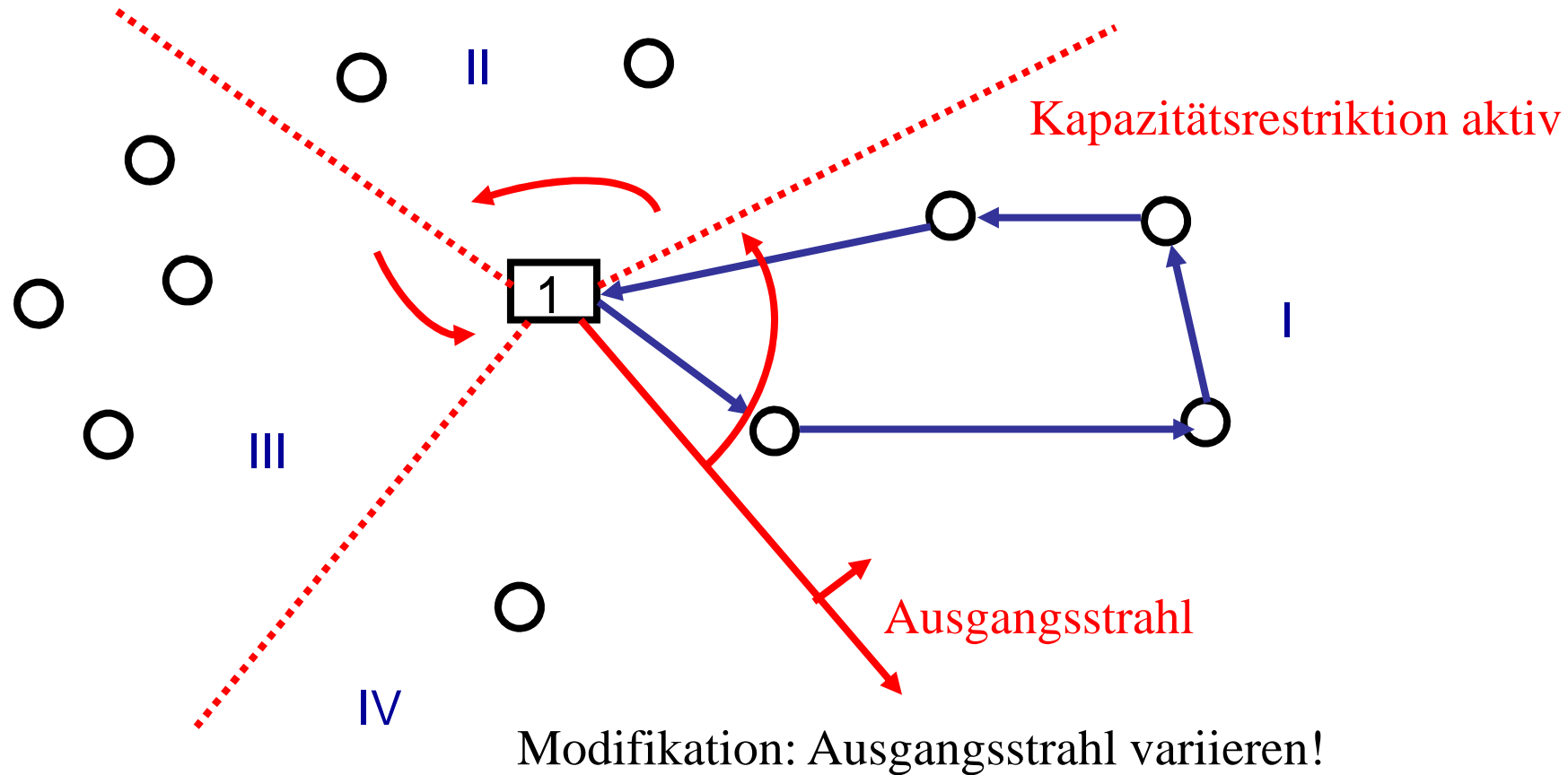
$$s_{II} = c_{1i} + c_{j1} - c_{ji} = 14$$

Idee:

- Initialisierung durch Pendeltouren
- Verbinden von Touren an den Tourenden, falls Savings entstehen und neue Tour zulässig (Restriktionen) ist.

3.3. VRP: Modellierung und heuristische Lösung

2. Das Sweep-Verfahren



3.3. VRP: Modellierung und heuristische Lösung

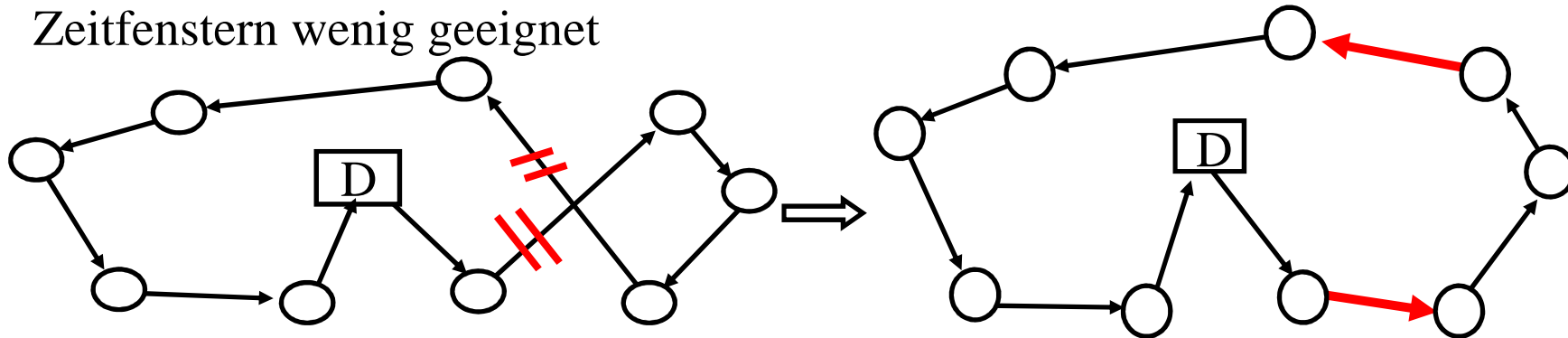
3.3.3. Verbesserungsverfahren für TSP und VRP (Bausteine einer Metaheuristik)

Verfahren	Nachbarschaft	Bemerkung	Eignung für das VRP
k-opt	k Kantenaustausche		wenig
Or-opt	Verschieben von 1-3 benachbarten Knoten	Teilmenge von 3-opt	gut
2-opt*	Austausch von Touranfangsstücken (bzw. Tourendstücken) zwischen 2 Touren	Teilmenge von 2-opt, nur für je 2 Touren	gut
CROSS	Austausch je eines beliebigen Tourabschnittes zwischen 2 Touren	enthält 2-opt*- und Or-opt-Nachbarschaft	gut
λ -Interchange	Austausch von Knotenteilmengen von maximal λ Knoten zwischen 2 Touren	enthält die Or-opt-Nachbarschaft	gut

3.3. VRP: Modellierung und heuristische Lösung

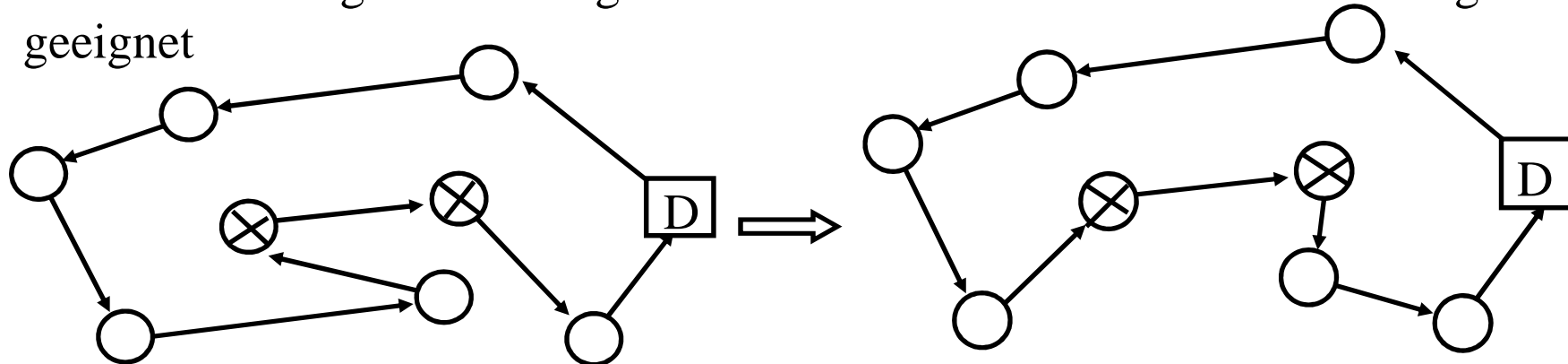
2-opt

In einem Teil der Tour wird die Orientierung umgekehrt → für Probleme mit Zeitfenstern wenig geeignet



Or-opt

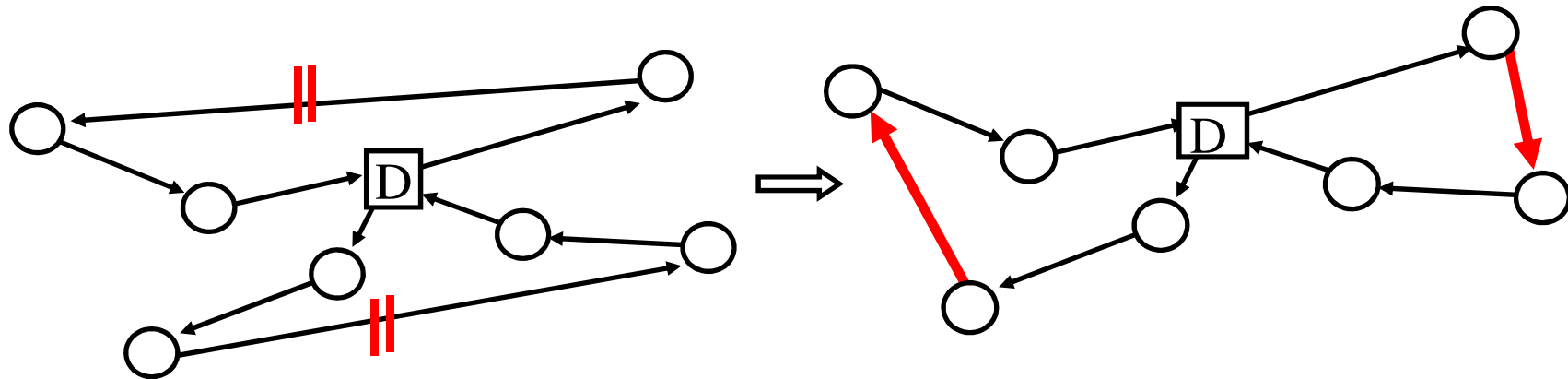
Die Orientierung wird nicht geändert → für Probleme mit Zeitfenstern gut geeignet



3.3. VRP: Modellierung und heuristische Lösung

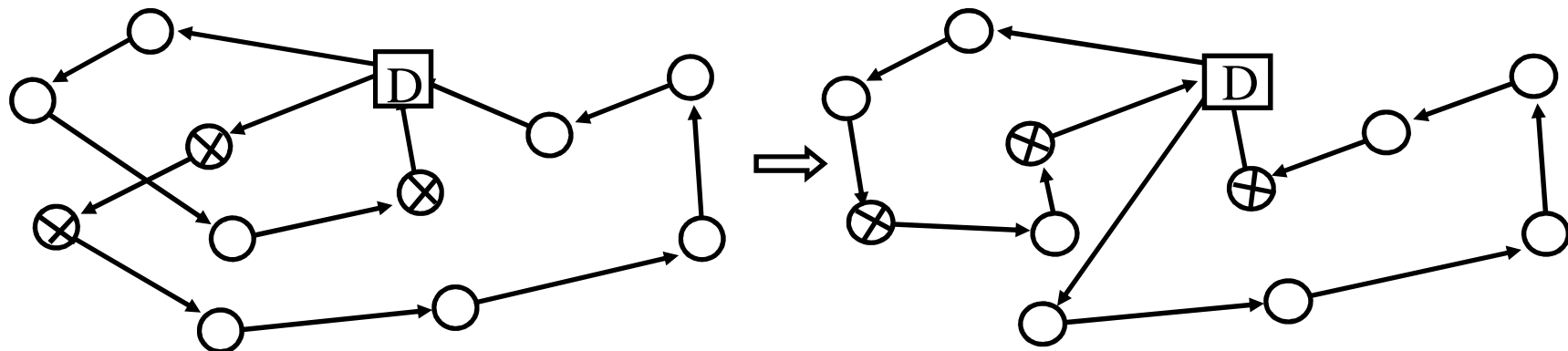
2-opt*

- Orientierungen von Teiltouren werden nicht geändert
- Kunden am Touranfang bleiben am Anfang, Kunden am Tourende bleiben am Tourende → für Probleme mit Zeitfenstern gut geeignet



Lambda-Interchange

- $\lambda=2$: 0,1 oder 2 Knoten aus Tour 1 in Tour 2 schieben und gleichzeitig 0,1 oder 2 Knoten aus Tour 2 in Tour 1 schieben
- z.B.: Schiebe einen Knoten aus Tour 1 in Tour 2, schiebe zwei Knoten aus Tour 2 und in Tour 1



4. Nichtlineare Optimierung

4.1. Grundlegende Begriffe und Eigenschaften

→ Übung

4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen

Konvexität von $f(x)$ nicht nur lokal (in $U(x^*)$) gegeben, sondern global auf ganz $M \subseteq \mathbb{R}^n$

$\Rightarrow x^*$ ist globales Minimum.

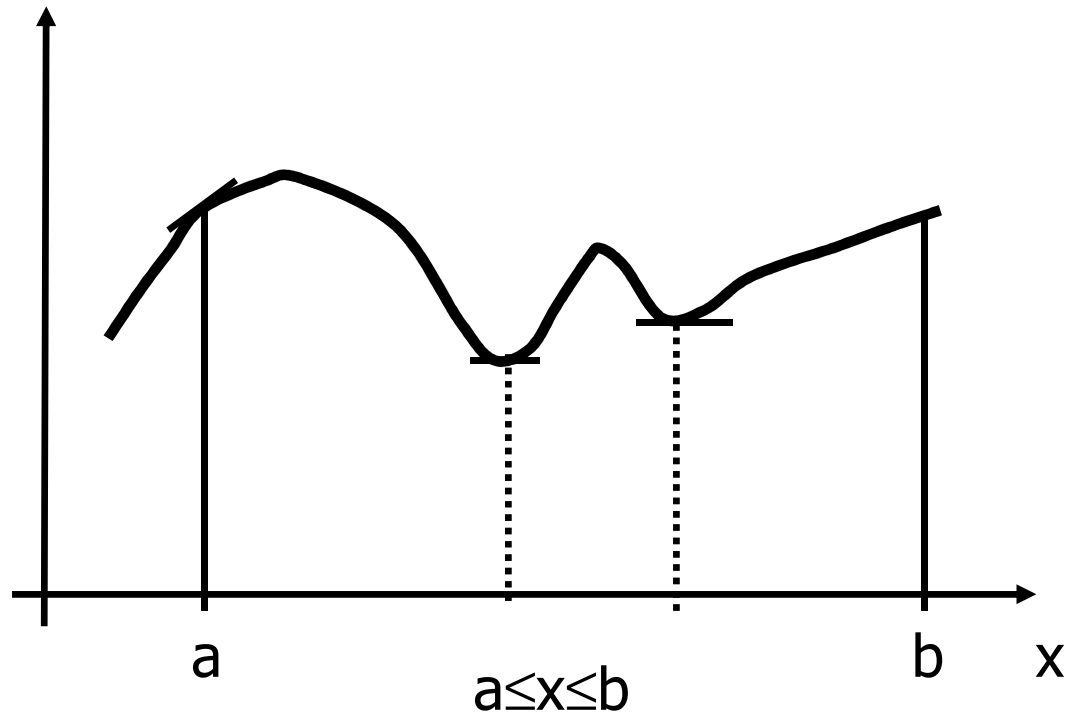
$\text{Min } f(x), \quad x \in M$

Satz 1:

Ist $f: \mathbb{R}^n \rightarrow \mathbb{R}^1$ auf $M \subseteq \mathbb{R}^n$ konvex, dann ist jedes (lokale) Minimum von f gleich dem globalen Minimum von f auf M .

Ist $a \in \mathbb{R}^1$ und $f: \mathbb{R}^n \rightarrow \mathbb{R}^1$ konvex $\Rightarrow M_a = \{x \mid x \in \mathbb{R}^n \wedge f(x) \leq a\}$ ist konvex.

4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen



Beweis:

$x^* \in M$ sei lokales Minimum von f

zu zeigen: x^* ist globales Minimum $\sim \forall x \in M: f(x^*) \leq f(x)$

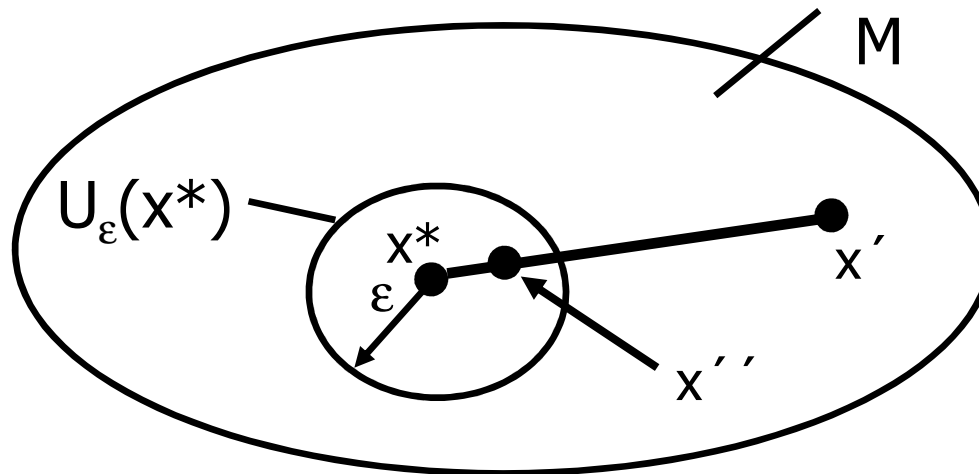
4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen

indirekt: x' sei „besser“, d.h. $f(x') < f(x^*)$

Da $x^* \in M$ lokales Minimum ist, gilt:

$\exists U_\varepsilon(x^*)$ mit $\varepsilon > 0$, so dass $\forall x \in U_\varepsilon(x^*) \cap M: f(x) \geq f(x^*)$.

Wegen Konvexität von M liegt Verbindungsgerade x^* nach x' ganz in M .



4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen

Betrachte x'' auf der Verbindungsgeraden mit $x'' \in U_\varepsilon(x^*)$, d.h.

$$\exists \lambda \in \left(0, \frac{\varepsilon}{|x^* - x'|} \right)$$

mit $x'' = (1-\lambda)x^* + \lambda x'$ und $x'' \in U_\varepsilon(x^*) \cap M$, d.h. $f(x'') \geq f(x^*)$.

- f konvex auf M und $f(x') < f(x^*)$:

$$\begin{aligned} f(x'') &= f((1-\lambda)x^* + \lambda x') \\ &\leq (1-\lambda)f(x^*) + \lambda f(x') \\ &< (1-\lambda)f(x^*) + \lambda f(x^*) \\ &= f(x^*) \end{aligned}$$

Widerspruch!

4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen

<u>Problem:</u>	$\min f(x)$ $x \in \mathbb{R}^n$ so dass $g_i(x) \leq 0 \quad \forall i=1, \dots, m$	Standardproblem der NLP
-----------------	--	-------------------------------

Zur Plausibilitätsbetrachtung seien f und g_i differenzierbar

→ \exists Gradient $\nabla f(x^*)$ von f im Punkt x^* mit:

1. Vektor senkrecht auf der Tangentialhyperebene in x^* an

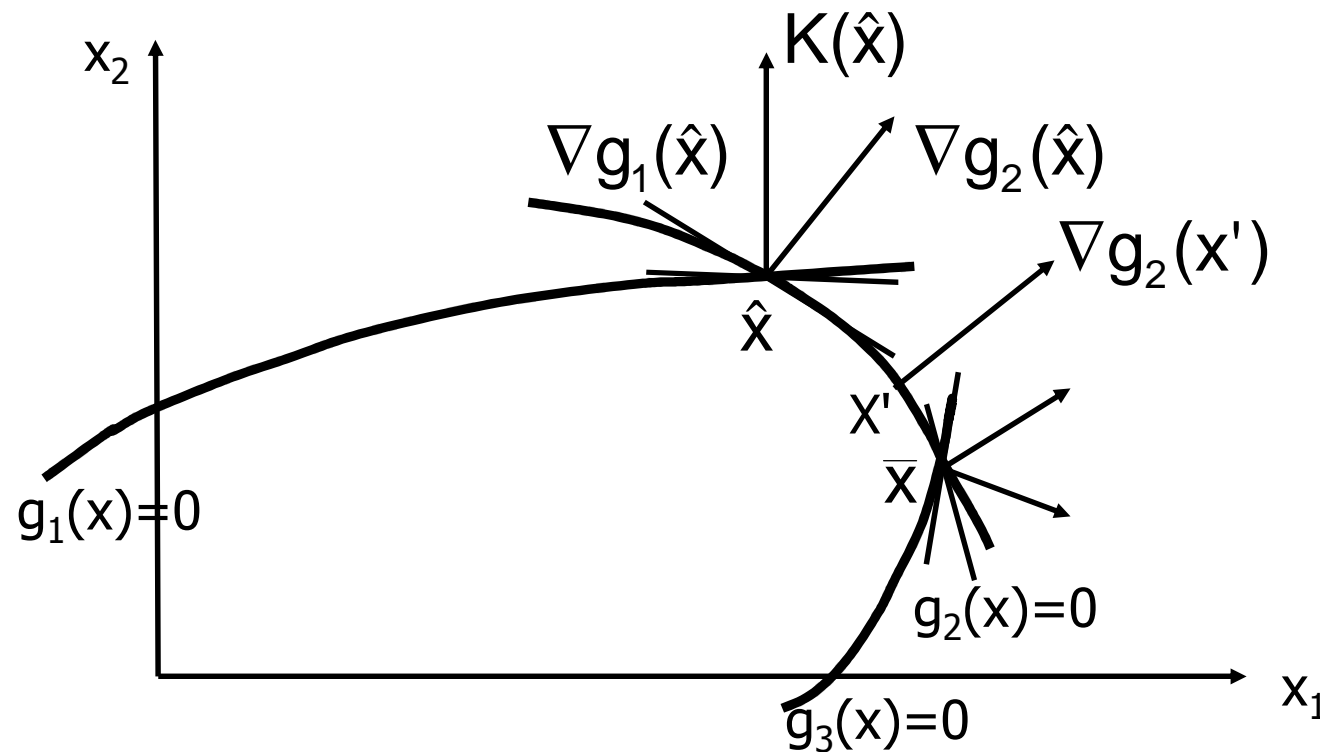
$$\{ x \mid f(x) = f(x^*) \}$$

2. $f(x)$ wächst in Richtung des Gradienten

→ für „ \leq -Restriktionen“ gilt: → In einem Punkt x^* mit $g_i(x^*)=0$ zeigt

$\nabla g_i(x^*)$ aus Z hinaus.

4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen



Lokale Optimalität eines Punktes x^* bedeutet:

- x^* zulässig
- x^* muss in gewisser Umgebung der beste Punkt sein
→ Ist $x^* \in \text{Rand}(Z) \rightarrow$ Jede verbessernde Richtung muss aus Z hinaus zeigen

4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen

„Verbessernde Richtung“

1) Richtung des steilsten Abstiegs in x^* ist $-\nabla f(x^*)$.

Jede verbessernde Richtung bildet mit $-\nabla f(x^*)$ einen Winkel $< 90^\circ$.

2) Liegt $-\nabla f(x^*)$ in dem Kegel $K(x^*)$, der durch die Gradienten der in x^* aktiven Restriktionen aufgespannt wird, dann zeigt jede verbessernde Richtung aus Z hinaus.

3) Sei I die Menge der Indizes der in x^* aktiven Restriktionen \rightarrow

$$K(x^*) = \{d \in \mathbb{R}^n \mid d = \sum_{i \in I} u_i \cdot \nabla g_i(x^*), u_i \geq 0\}$$

4) $-\nabla f(x^*)$ liegt genau dann in $K(x^*)$, wenn gilt: $\forall i, i \in I (\exists u_i \geq 0)$ mit:

$$-\nabla f(x^*) = \sum_{i \in I} u_i \cdot \nabla g_i(x^*)$$

5) Setze zur Vereinfachung der Schreibweise: $I = I(x^*)$

4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen

Kuhn-Tucker-Bedingungen

$$(KTB) \quad \begin{cases} \nabla f(x^*) + \sum_{i \in I} u_i \cdot \nabla g_i(x^*) = 0 \\ u_i \geq 0 & \forall i \in I \\ g_i(x^*) \leq 0 & \forall i = 1, \dots, m \end{cases}$$

Satz 1: (Hinreichende KTB)

Seien $f: \mathbb{R}^n \rightarrow \mathbb{R}$ und $g_i: \mathbb{R}^n \rightarrow \mathbb{R} \quad \forall i = 1, \dots, m$ stetig differenzierbare und konvexe Funktionen und das NLP sei: $\min f(x)$, so dass $g_i(x) \leq 0 \quad \forall i$

Dann gilt:

Gibt es in einem Punkt $x^* \in \mathbb{R}^n$ für alle aktiven Restriktionen $i \in I$ nichtnegative Skalare $u_i \geq 0$, so dass die KTB erfüllt sind, dann ist x^* globales Minimum des NLP.

4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen

Notwendige Kuhn-Tucker-Bedingungen:

Konvexität wird nicht verlangt, aber Regularitätsannahme (lineare Unabhängigkeit der Gradienten der in x^* aktiven Restriktionen)

→ Zu jedem lokalen Minimum kann man eine Lösung der KTB finden.

Satz 2: (Notwendige KTB)

Seien $f: \mathbb{R}^n \rightarrow \mathbb{R}$ und

$g_i: \mathbb{R}^n \rightarrow \mathbb{R} \quad \forall i=1, \dots, m$ stetig differenzierbare Funktionen

$x^* \in \mathbb{R}^n$, $I = \{i \in N \mid g_i(x^*) = 0\}$, $\nabla g_i(x^*)$ linear unabhängig $\forall i \in I$, dann gilt:

Ist x^* ein lokales Minimum des NLP, dann gibt es $u_i \geq 0 \quad \forall i \in I$, so dass die KTB erfüllt sind.

4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen

Beweis:

- Trennungssätze konvexer Kegel

Speziell:

a) x^* innerer Punkt von Z .

$$\rightarrow I = \emptyset$$

$$\rightarrow \text{KTB: } \nabla f(x^*) = 0, \quad g_i(x^*) < 0 \quad \forall i=1, \dots, m$$

b) x^* liegt auf genau einer Hyperfläche,

$$\text{KTB: } g_{i_0}(x) = 0 \quad \text{d.h. } I = \{i_0\}$$

$$\nabla f(x^*) + u_{i_0} \cdot \nabla g_{i_0}(x^*) = 0$$

$$u_{i_0} \geq 0$$

$$g_i(x^*) \leq 0 \quad \forall i=1, \dots, m$$

4.2. Konvexe Optimierungsprobleme und Kuhn-Tucker-Bedingungen

Kritik:

Sucht man (konstruktiv) nach Punkten, die die KTB erfüllen sollen, weiß man nicht, ob / welche Restriktionen in derartigen Punkten aktiv sind.

→ modifizierte Form der KTB: $(KTB)'$ (äquivalent zu KTB)

$$\nabla f(x^*) + \sum_{i=1}^m u_i \cdot \nabla g_i(x^*) = 0$$

$$u_i \cdot g_i(x^*) = 0 \quad \forall i=1, \dots, m$$

$$u_i \geq 0 \quad \forall i=1, \dots, m$$

$$g_i(x^*) \leq 0 \quad \forall i=1, \dots, m$$

Bemerkung:

Ein Punkt (x^*, u^*) der die KTB erfüllt, kann auch als ein Sattelpunkt der Lagrangefunktion

$$L(x, u) = f(x) + \sum_{i=1}^m u_i \cdot g_i(x) \quad (=F(x, u)) \text{ interpretiert werden.}$$

4.3. Lagrangefunktion und Optimalitätsbedingungen

Modell M1:

$$\begin{array}{ll} \min & f(x) \\ \text{s.d.} & g_i(x) \leq 0, \quad i=1,\dots,m \\ & x \geq 0 \end{array}$$

Def.1: Die Funktion $L(x,u) = f(x) + u^T \cdot g(x)$
mit: $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $g(x)^T = (g_1(x), \dots, g_m(x))$ heißt
Lagrange-Funktion zu M1

Def. 2: Ein Vektor $(x_0, u_0)^T \in \mathbb{R}^{n+m}$ mit $x_0 \geq 0$, $u_0 \geq 0$ heißt
Sattelpunkt von $L(x,u)$, wenn $\forall x \in \mathbb{R}^n, x \geq 0 \wedge \forall u \in \mathbb{R}^m, u \geq 0$
 $L(x_0, u) \leq L(x_0, u_0) \leq L(x, u_0)$ gilt.

4.3. Lagrangefunktion und Optimalitätsbedingungen

Satz 1: (Notwendige KTB)

$f(x)$ und $g_i(x)$ seien für alle $i=1,\dots,m$ partiell stetig differenzierbar.

L_x, L_u bezeichnen die Gradienten von L bzgl. x bzw. u .

Notwendige Bedingung für einen SP von L in $(x_0, u_0)^T$:

$$L_x(x_0, u_0) \geq 0 \quad x_0^T \cdot L_x(x_0, u_0) = 0$$

$$L_u(x_0, u_0) \leq 0 \quad u_0^T \cdot L_u(x_0, u_0) = 0$$

$$u_0 \geq 0, \quad x_0 \geq 0$$

Dabei sind:
$$L(x, u) = f(x) + \sum_{i=1}^m u_i \cdot g_i(x)$$

$$L_x = \nabla f(x) + \sum_{i=1}^m u_i \cdot \nabla g_i(x)$$

$$L_u = g(x) = (g_1(x), \dots, g_m(x))^T$$

4.3. Lagrangefunktion und Optimalitätsbedingungen

Satz 2:

Ist $(x_0, u_0)^T$, $x_0, u_0 \geq 0$ ein SP von L , so ist x_0 eine optimale Lösung von Modell $M1$.

Beweis:

Def. SP \Rightarrow Wegen $L(x_0, u) \leq L(x_0, u_0) \quad \forall u \in \mathbb{R}^m, u \geq 0$ gilt

$$f(x_0) + u^T \cdot g(x_0) \leq f(x_0) + u_0^T \cdot g(x_0) \quad (\diamond)$$

$$\Rightarrow u^T \cdot g(x_0) \leq u_0^T \cdot g(x_0) \quad \forall u \in \mathbb{R}^m, u \geq 0.$$

Das ist nur möglich, wenn $g(x_0) \leq 0$ ist (d.h. x_0 zulässig für $M1$).

$$\Rightarrow u_0^T \cdot g(x_0) \leq 0$$

$$\text{In } (\diamond) \text{ } u=0 \text{ setzen: } \Rightarrow u_0^T \cdot g(x_0) \geq 0$$

$$\Rightarrow u_0^T \cdot g(x_0) = 0$$

4.3. Lagrangefunktion und Optimalitätsbedingungen

Zweite Ungleichung in (SP)-Def.

$$\Rightarrow f(x_0) + u_0^T g(x_0) \leq f(x) + u_0^T g(x) \quad \forall x \in \mathbb{R}^n, x \geq 0$$

$$\text{Wegen } u_0^T g(x_0) = 0 \quad \Rightarrow \quad f(x_0) \leq f(x) + u_0^T g(x)$$

Für alle zulässigen x für MI gilt $g(x) \leq 0$

$$\Rightarrow f(x_0) \leq f(x) \quad \forall x \in \mathbb{R}^n \text{ und } x_0 \text{ zulässig, d.h. optimale Lösung.}$$

Bemerkung:

Konvexität von $L(x,u)$ in x und Konkavität von $L(x,u)$ in u bisher nicht vorausgesetzt.

Setzt man dies zusätzlich voraus, sind die KTB **NHB** für einen SP von L .

Umkehrung von Satz 2

Erfordert zusätzlich zur Konvexität eine zusätzliche Bedingung (*Slater-Bedingung*).

4.3. Lagrangefunktion und Optimalitätsbedingungen

Def. 3:

In $M1$ sei $I = \{1, 2, \dots, m\}$ derart disjunkt in $I_1 \cup I_2$ aufgespalten, dass I_2 die Indexmenge der linearen Restriktionen bezeichnet.

Falls ein $x \in \mathbb{R}^n$ derart existiert, dass $\forall i \in I_1$ gilt $g_i(x) < 0$, so erfüllt x die Slater-Bedingung.

(x ist innerer Punkt bzgl. der nichtlinearen Restriktionen).

Satz 3:

Erfüllt $M1$ (einschließlich Konvexitätsvoraussetzungen) die Slater-Bedingung, so sind die KTB NHB für eine optimale Lösung von $M1$.

Nützlichkeit der KTB

- Überprüfung einer Kandidatenlösung auf Optimalität
- Grundlage für Theorie (z.B. Dualität)
- In Spezialfällen zur konstruktiven Bestimmung optimaler Lösungen verwendbar, z.B. Quadratisches Programmieren

4.4. Quadratische Probleme und das Verfahren von Wolfe

Modell M2:

$$\begin{array}{ll} \min & f(x) = c^T x + \frac{1}{2} x^T Q x \\ \text{s.d.} & Ax \leq b, \text{ (falls } \neq \emptyset, \text{ konv. Polyeder)} \\ & x \geq 0 \\ & c, x \in \mathbb{R}^n, A = A_{m,n} \end{array}$$

Q symmetrisch und positiv semidefinit. ($\rightarrow f(x)$ konvex).

$$\rightarrow: L(x, u) = c^T x + \frac{1}{2} x^T Q x + u^T \cdot (Ax - b)$$

L konvex in x und konkav in u .

\rightarrow KTB sind NHB für optimale Lösung, Slater-Bedingung nicht relevant.

4.4. Quadratische Probleme und das Verfahren von Wolfe

KTB:

$$\left. \begin{array}{l} L_x(x, u) = c + Qx + A^T u \geq 0 \\ L_u = Ax - b \leq 0 \end{array} \right\} (\alpha)$$

$$L_x \cdot x_0 = 0, \quad L_u \cdot u_0 = 0 \quad (\beta)$$

Hinzufügen von Schlupfvariablen zu $(\alpha) \rightarrow \text{NHB}$:

$$\boxed{\left. \begin{array}{l} c + Qx + A^T u - s_1 = 0 \\ -b + Ax + s_2 = 0 \end{array} \right\} (\gamma)}$$

unter Beachtung von (β)

\rightarrow ***Algorithmus von Wolfe (1959)***

Zulässige Lösung von (γ) mit Hilfe der Simplexmethode
(unter Beachtung von (β)) zu finden.

Formuliere die KTB für Modell M2	
Füge linearen Bedingungen Schlupfvariable hinzu	
Liegt eine zulässige Basislösung unter Beachtung von (β) vor?	
Ja	Nein
Lösung optimal	<p>Füge, wo notwendig, Hilfsvariablen zur Erlangung einer Ausgangslösung von (γ) und (β) hinzu</p> <p>Iteriere mit M-Methode, bis zulässige Lösung erreicht oder feststeht, dass Lösungsraum leer ist.</p> <p>Beachte dabei (β) durch beschränkten Basiseintritt: Ist x_j in Lösung, darf s_{1j} nicht aufgenommen werden und umgekehrt.</p>

Algorithmus von Wolfe

Beispiel:

minimiere $z = -20x_1 + 10x_2 + 3x_1^2 + 2x_2^2$

so dass $2x_1 - x_2 \leq 6$

$-x_1 + x_2 \leq 10$

$2x_1 + 3x_2 \geq 8$

$x_1, x_2 \geq 0$

1. Schritt

Es ist: $c^T = (-20, 10)$; $b^T = (6, 10, -8)$

$$Q = \begin{pmatrix} 6 & 0 \\ 0 & 4 \end{pmatrix} \quad A = \begin{pmatrix} 2 & -1 \\ -1 & 1 \\ -2 & -3 \end{pmatrix}$$

Die Kuhn-Tucker-Bedingungen lauten also:

$$L_x = \begin{cases} -20 + 6x_1 + 2u_1 - u_2 - 2u_3 \geq 0 \\ 10 + 4x_2 - u_1 + u_2 - 3u_3 \geq 0 \end{cases}$$
$$L_u = \begin{cases} -6 + 2x_1 - x_2 \leq 0 \\ -10 - x_1 + x_2 \leq 0 \\ +8 - 2x_1 - 3x_2 \leq 0 \end{cases}$$

$$L_x \cdot x_0 = 0$$

$$L_u \cdot u_0 = 0$$

4.4. Quadratische Probleme und das Verfahren von Wolfe

2. Schritt

Hinzufügen von Schlupfvariablen und Umordnen des Systems:

$$\begin{array}{rcl} 6x_1 & + 2u_1 - u_2 - 2u_3 & - s_{11} = 20 \\ & -4x_2 + u_1 - u_2 + 3u_3 & + s_{12} = 10 \\ 2x_1 - x_2 & & + s_{21} = 6 \\ -x_1 + x_2 & & + s_{22} = 10 \\ 2x_1 + 3x_2 & & -s_{23} = 8 \\ x_j, u_i, s_{ij} \geq 0, & j = 1,2; & i = 1,2,3 \end{array}$$

3. Schritt:

Eine zulässige Lösung liegt noch nicht vor. Daher Hinzufügen von Hilfsvariablen h_1 und h_2 . Damit ergibt sich folgendes Tableau:

⇒ weiter in der Übung

4.5. Dualität in der Nichtlinearen Optimierung

$$\begin{aligned} \text{(M1)} \quad & \min f(x) \\ & \text{so daß } g_i(x) \leq 0, i = 1, \dots, m \\ & x \in \mathbb{R}^n \end{aligned}$$

(NNB werden, falls sie existieren, unter die g_i subsummiert)

Lagrange Funktion $L: \mathbb{R}^{n+m} \rightarrow \mathbb{R}^1$

wie bisher:

$$L(x, u) = f(x) + \sum_{i=1}^m u_i g_i(x) = f(x) + u^T g(x)$$

Lagrange Multiplikatoren: u_1, u_2, \dots, u_m

4.5. Dualität in der Nichtlinearen Optimierung

(\bar{x}, \bar{u}) mit $\bar{u} \geq 0$ heißt **Sattelpunkt von L** (SP von L), wenn

$$L(\bar{x}, u) \leq L(\bar{x}, \bar{u}) \leq L(x, \bar{u})$$

Für alle $x \in \mathbb{R}^n$, $u \in \mathbb{R}^{m+}$ ($u \geq 0$) gilt.

KTB: Sind f und alle g_i konvex und ist die Slaterbedingung erfüllt, so ist $\bar{x} \in \mathbb{R}^n$ genau dann eine optimale Lösung, falls $\bar{u} \in \mathbb{R}^{m+}$ derart existiert, daß (\bar{x}, \bar{u}) Sattelpunkt von L ist.

Zusammenhang zwischen Existenz eines SP von L und Bedingungen, die den KTB entsprechen, ohne stetige Differenzierbarkeit und Konvexität von f , g_1 , g_2, \dots, g_m vorauszusetzen.

Satz 1:

(\bar{x}, \bar{u}) ist genau dann SP von L, wenn gilt:

$$L(\bar{x}, \bar{u}) = \min_{x \in \mathbb{R}^n} L(x, \bar{u}) \quad (1)$$

$$\bar{u}^T g(\bar{x}) = 0$$

$$g(\bar{x}) \leq 0$$

$$\bar{u} \geq 0$$

(Bis auf (1) Übereinstimmung mit den KTB)

→ Einführung eines dualen Optimierungsproblems

4.5. Dualität in der Nichtlinearen Optimierung

Es sei

$$\begin{aligned} F(x) &:= \sup_{u \in \mathbb{R}^{m+}} L(x, u) = \sup_{u \in \mathbb{R}^{m+}} (f(x) + u^T g(x)) \\ &= \begin{cases} f(x), & \text{falls } g(x) \leq 0 \\ \infty, & \text{falls } g(x) > 0. \end{cases} \end{aligned}$$

Deshalb kann man das **primale Problem** schreiben als:

$$\left. \begin{array}{l} \text{Minimiere } F(x) := \sup_{u \in \mathbb{R}^{m+}} L(x, u) \\ \text{s.d. } x \in \mathbb{R}^n \end{array} \right\} \quad (2)$$

Dazu **duales Problem:**

Maximiere $G(u) := \inf_{x \in R^n} L(x, u)$

s.d. $u \in R^{m+}$

Lagrangemultiplikatoren u_1, u_2, \dots, u_m (den m NB zugeordnet) spielen die Rolle der Dualvariablen.

Verallgemeinerung

Anstelle $x \in R^n$:

$x \in X \subseteq R^n$ (X kann eine diskrete Menge sein; alle Gitterpunkte oder endliche Menge des R^n)

4.5. Dualität in der Nichtlinearen Optimierung

(P)	{	Minimiere $F(x) := \sup_{u \in \mathbb{R}^{m+}} L(x, u)$
primales Problem		s.d. $x \in X \subseteq \mathbb{R}^n$
(\bar{P})	{	Maximiere $G(u) := \inf_{x \in X} L(x, u)$
duales Problem		s.d. $u \in \mathbb{R}^{m+}$

Satz 2:

Die Zielfunktion G des dualen Problems (\bar{P}) ist konkav auf \mathbb{R}^{m+} .

Beweis: Neumann/Morlock S. 580

Bedeutung:

Die Funktion G ist konvex und (\bar{P}) ein konvexes opt. Problem ohne Konvexitätsvoraussetzungen an f, g_1, g_2, \dots, g_m und an X , d.h. eine opt. Lösung des DP (\bar{P}) kann in der Regel einfacher bestimmt werden als eine solche für das i.a. nicht-konvexe PP (P) .

Dualitätssätze

Satz 3: (Analogon zum LP)

Sind \bar{x} eine zulässige Lösung von (P) und \bar{u} eine zulässige Lösung von (\bar{P}) , so gilt: $F(\bar{x}) \geq G(\bar{u})$.

Beweis:

$$F(\bar{x}) = \sup_{u \in \mathbb{R}^{m+}} (f(\bar{x}) + u^T g(\bar{x})) \geq f(\bar{x}) + \bar{u}^T g(\bar{x}) \geq \inf_{x \in X} (f(x) + \bar{u}^T g(x)) = G(\bar{u}).$$

Folgerungen

- (1) Sind \bar{x} und \bar{u} zulässige Lösungen von (P) bzw. (\bar{P}) und gilt $F(\bar{x}) = G(\bar{u})$:
→ dann stellen \bar{x} und \bar{u} optimale Lösungen von (P) bzw. (\bar{P}) dar.
- (2) Ist F auf X nicht nach unten bzw. G auf \mathbb{R}^{m+} nicht nach oben beschränkt, dann besitzt (\bar{P}) bzw. (P) keine zulässige Lösung.

**Satz 4: (Zusammenhang zwischen Gleichheit der Optimalwerte für
(P) und (\bar{P}) und Existenz eines Sattelpunktes von L)**

L besitzt genau dann einen SP (x^*, u^*) , wenn x^* eine opt. Lösung von (P) und u^* eine opt. Lösung von (\bar{P}) sind und wenn $F(x^*) = G(u^*)$ gilt.

(Für $X \subset \mathbb{R}^n$ in SP-Definition $x \in \mathbb{R}^n$ durch $x \in X$ ersetzen.)

-Hat L keinen SP, gilt $F(x^*) > G(u^*)$ für opt. Lösung x^* von (P) bzw. u^* von (\bar{P}) .

4.5. Dualität in der Nichtlinearen Optimierung

$F(x^*) - G(u^*)$ heißt „**Dualitätslücke**“.

Eine solche Dualitätslücke kann im Unterschied zum LP auch für $X = \mathbb{R}^n$ auftreten.

Dualitätssatz:

Sind f, g_1, g_2, \dots, g_m konvex auf der konvexen Menge X , ist die Slaterbedingung erfüllt und existiert eine opt. Lösung von (P), dann gilt:

- (a) (\bar{P}) besitzt eine opt. Lösung u^*
- (b) $F(x^*) = G(u^*)$ (keine Dualitätslücke)
- (c) Sind u^* eine opt. Lösung von (\bar{P}) und x^+ ein globaler Minimalpunkt von $L(\bullet, u^*)$ auf X mit $g(x^+) \leq 0$ und $u^{*T} g(x^+) = 0$, so ist x^+ eine opt. Lösung von (P).

Nützlichkeit von (c):

- Bei bekannter Lösung u^* des DP (\bar{P}) eine opt. Lösung von (P) bestimmen.

z.B. günstig, falls $L(\bullet, u^*)$ auf X streng konvex ist

(gilt z.B. falls f oder eine zu positivem u_i^* gehörende Funktion g_i streng konvex ist)

→ $L(\bullet, u^*)$ hat genau einen lokalen und globalen Minimalpunkt x^+ auf X , der opt. Lösung von (P) ist.

(d.h. $g(x^+) \leq 0$ und $u^{*T}g(x^+) = 0$ sind automatisch erfüllt).

- Optimale Lösung u^* von (\bar{P}) mit Subgradientenverfahren ermitteln:
(- $g(\bar{x})$ ist Subgradient von $-G$ an der Stelle u , falls für $u \in \mathbb{R}^{m+}$
 $\bar{x} \in X$ ein Minimalpunkt von $L(\cdot, u)$ auf X ist.)

4.6. Ausgewählte numerische Verfahren

4.6.1 Minimierung ohne Restriktionen (Unconstrained Optimization)

Prinzipielles Vorgehen

Problem: $\min f(x)$

gegeben sei: $x^k \in \mathbb{R}^n$ d.h. zulässiger Iterationspunkt

gesucht: $x^{k+1} \in \mathbb{R}^n$

mit: $f(x^{k+1}) < f(x^k)$

Iteration besteht (meist) aus zwei Schritten:

1. Bestimme eine Richtung $d^k \in \mathbb{R}^n$, in der f fällt (ZF verbessert!)
2. Löse $\min\{f(x^k + \mu d^k) \mid \mu > 0\}$ (Strahloptimierung, Schrittweitenbestimmung)
Setze: $x^{k+1} = x^k + \mu^k d^k$; μ^k - optimale Schrittweite

4.6.2 Mehrdimensionales Suchen - Differenzierbarkeit vorausgesetzt

a) Steilster Abstieg: Gradientenverfahren

$\nabla f(x)$ zeigt in Richtung des stärksten Anstiegs von $f(x)$

→ Richtung des steilsten Abstiegs im Punkt x^k ist: $-\nabla f(x^k)$

→ Methode „Gradientenverfahren“

- gegeben: x^k - k-ter Iterationspunkt

$$d^k = -\nabla f(x^k)$$

- bestimme: $\mu^k = \arg \min \{f(x^k + \mu d^k) \mid \mu \geq 0\}$ $x^{k+1} = x^k + \mu^k d^k$

- Abbruchbedingung: $|\nabla f(x^k)| < \varepsilon$

Für die erzeugte Punktfolge gilt: $f(x^{k+1}) < f(x^k)$

(Häufig schlechte Konvergenz in der Nähe des Optimums, „Zick-Zack-Kurs“)

4.6. Ausgewählte Numerische Verfahren

b) Mehrdimensionales Newton-Verfahren

jetzt: $f: \mathbb{R}^n \rightarrow \mathbb{R}^1$ sei zweimal stetig differenzierbar

→ Quadratische Näherung von $f(x)$ in x^k (mehrdim. Taylorentwicklung)

$q(x) = f(x^k) + \nabla f(x^k)^T(x-x^k) + \frac{1}{2}(x-x^k)^T H(x^k)(x-x^k)$ mit H : Hesse-Matrix

Analoges Vorgehen wie im eindimensionalen Fall:

$$\nabla q(x) = 0 \Leftrightarrow \nabla f(x^k) + H(x^k)(x-x^k) = 0$$

Ist $H(x)$ invertierbar, so gilt: $x^{k+1} = x^k - H^{-1}(x^k)\nabla f(x^k)$

Konvergenzbetrachtungen

1. Zur Idee des Newton-Verfahrens

Quadratisches Problem $f(x) = \frac{1}{2}x^T A x - b^T x + c$

→ Ist lösbar in einem Iterationsschritt, wenn man

$x^* := A^{-1}b = x^\circ - A^{-1}(Ax^\circ - b)$ (*) für ein beliebiges $x^\circ \in \mathbb{R}^n$ setzt

A ist dabei als positiv definit vorausgesetzt.

Analogieüberlegung:

$$\left. \begin{array}{l} \nabla f(x^\circ) = Ax^\circ - b \\ H(x^\circ) = \nabla^2 f(x^\circ) = A \end{array} \right\} \begin{array}{l} \text{allgemein} \\ x^{k+1} = x^k - [H(x^k)]^{-1} \nabla f_k \\ \nabla f_k = \nabla f(x_k) \end{array}$$

Vergleich mit (*) $d_k = -[H(x^k)]^{-1} \nabla f(x^k)$

allgemein $x^{k+1} = x^k + \mu^k d^k$

keine explizite Schrittweitenbestimmung in Optimumsnähe.

Konvergenzverhalten

Satz 1

Es sei x^* optimal für das unrestringierte Optimierungsproblem:

$$\nabla f(x^*) = 0$$

$$\nabla^2 f(x^*) = H(x^*) \text{ positiv definit}$$

4.6. Ausgewählte Numerische Verfahren

Falls x^0 hinreichend nahe bei x^* , konvergiert das Newton-Verfahren

$$x^{k+1} = x^k - [H(x^k)]^{-1} \nabla f(x^k)$$

quadratisch gegen x^* , d.h. es gilt:

$$\|x^{k+1} - x^*\| < \gamma \|x^k - x^*\|^2, \quad 0 < \gamma < 1$$

→ Vorteil: Quadratische Konvergenz

Nachteile:

- Notwendigkeit der Berechnung zweiter Ableitungen in jedem Iterationsschritt
 - Lösung des Gleichungssystems $H(x^k)d_k = -\nabla f(x^k)$ ist aufwendig
 - $H(x^k)$ kann singulär oder schlecht konditioniert sein
 - Verfahren nur lokal konvergent
- Quasi-Newton-Verfahren

4.6.3 Verfahren zulässiger Richtungen

- Explizite Berücksichtigung der Nebenbedingungen im Verlaufe des Optimierungsverfahrens
- Prinzipielles Vorgehen:

Problem: $\min_{x \in Z} f(x)$

$x^k \in Z \subset \mathbb{R}^n$ sei gegeben

Ziel: Verbesserung der zulässigen Lösung

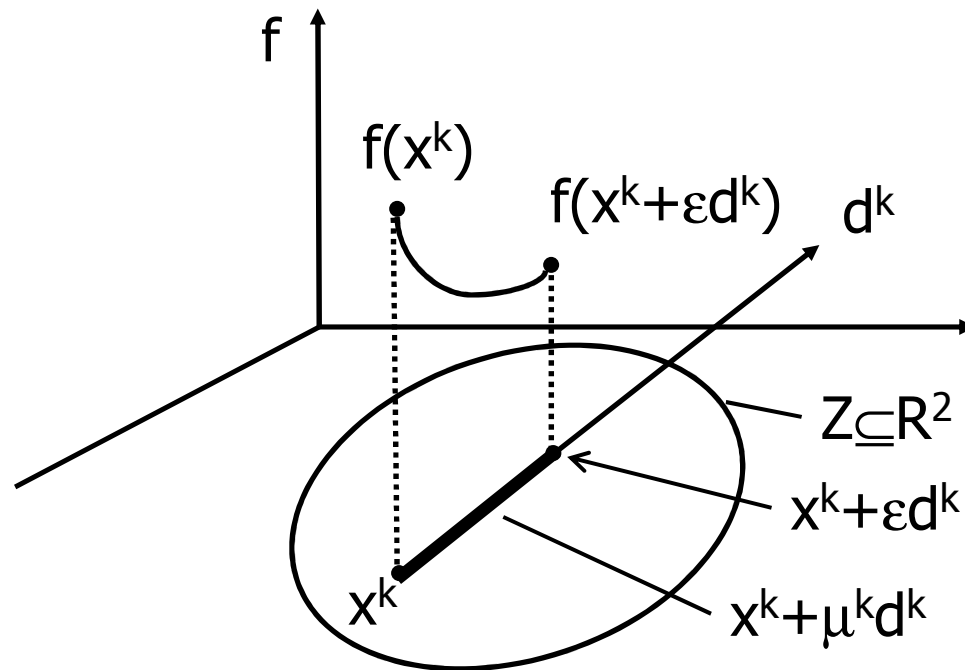
Vorgehen: Bestimme Vektor $d^k \in \mathbb{R}^n$ mit:

1. $x^k + \mu^k d^k \in Z \quad \forall \mu^k \in \mathbb{R}^1 \quad \text{mit} \quad 0 \leq \mu^k < \varepsilon$
2. $f(x^k + \mu^k d^k) < f(x^k) \quad \text{für alle } \mu^k \text{ wie in 1.}$

4.6. Ausgewählte Numerische Verfahren

d^k heißt „verbessernde zulässige Richtung“, denn:

- bewegt man sich von x^k in Richtung d^k , verbessert sich der ZF-Wert.
- geht man nicht zu weit, bleibt man zulässig



3. Bestimmung einer optimalen Schrittweite $\mu^k \in [0, \varepsilon]$
4. Festlegung des nächsten Iterationspunktes $x^{k+1} = x^k + \mu^k d^k$
 \exists unterschiedliche Methoden zur Bestimmung von d^k :
hier: Zoutendijk

Verfahren von Zoutendijk

Spezifikation des NLP:

$$\min_{x \in Z} f(x) \quad \text{mit: } Z = \{x \in \mathbb{R}^n \mid g_i(x) \leq 0 \quad \forall i=1, \dots, m\}$$

f, g_i differenzierbar

Satz 1:

Es sei $\bar{x} \in Z$, $I = \{i \in \{1, 2, \dots, m\} \mid g_i(\bar{x}) = 0\}$. Wenn $d \in \mathbb{R}^n$ die Ungleichungen $\nabla f(\bar{x})^T d < 0$ und $\nabla g_i(\bar{x})^T d < 0 \quad \forall i \in I$ erfüllt, dann ist d eine verbessernde zulässige Richtung.

4.6. Ausgewählte Numerische Verfahren

Für gegebenes \bar{x} lässt sich eine derartige Richtung d durch Lösung eines LP mit den Variablen d_j und z bestimmen.

$$\left. \begin{array}{ll} \min & z \\ \text{so dass} & \nabla f(\bar{x})^T \cdot d - z \leq 0 \\ & \nabla g_i(\bar{x})^T \cdot d - z \leq 0 \quad \text{für } i \in I \\ & -1 \leq d_j \leq +1 \quad \forall j=1, \dots, n \end{array} \right\} \text{LP}(\bar{x})$$

Die Bedingungen $-1 \leq d_j \leq +1$ schränken die Länge des Lösungsvektors d ein und verhindern unbeschränkte Lösungen.

Es gilt (ohne Beweis):

(\bar{z}, \bar{d}) sei eine Lösung von $\text{LP}(\bar{x})$.

Falls $\bar{z} < 0 \rightarrow \bar{d}$ ist eine verbessernde zulässige Richtung

Falls $\bar{z} = 0 \rightarrow \bar{x}$ ist ein Kuhn-Tucker-Punkt.

4.6.4 Minimierung mit Nebenbedingungen

Siehe auch Verfahren von Wolfe

Strafkosten- und Barriere-Verfahren

Nebenbedingungen derart in Zielfunktion einbeziehen, dass unzulässige Punkte Strafkosten verursachen. Lösen der unrestringierten Probleme mit Methoden aus dem vorangehenden Abschnitt.

- Penalty (Strafkosten)-Verfahren: $\{\mathbf{x}^k\}_{k=0,1,\dots}$ Folge unzulässiger Punkte (Näherung an den Lösungsraum „von außen“)
- Barriere-Verfahren: $\{\mathbf{x}^k\}_{k=0,1,\dots}$ Folge zulässiger Punkte $\rightarrow \mathbf{x}^k \in Z$ muss bekannt sein.

4.6. Ausgewählte Numerische Verfahren

Penalty-Verfahren

$$\underline{\text{NLP:}} \quad \min_{x \in Z} f(x) \quad \text{mit} \quad Z = \left\{ \begin{array}{l} x \in \mathbb{R}^n \mid g_i(x) \leq 0; \quad i=1, \dots, m \\ h_i(x) = 0; \quad i=1, \dots, k \end{array} \right\}$$

Strafkostenfunktion:

$$S(x) = \sum_{i=1}^m [\max\{0, g_i(x)\}]^p + \sum_{i=1}^k |h_i(x)|^p \quad \text{mit } 1 \leq p \leq N$$

Es gilt:

(i) $S(x) \geq 0$

(ii) $x \in Z \iff S(x) = 0$

Für zulässige Punkte, d.h. $x \in Z$, ist $S(x) = 0$. Je stärker ein Punkt Restriktionen verletzt, desto größer ist der Wert der Strafkostenfunktion.

4.6. Ausgewählte Numerische Verfahren

Axiomatik:

Mit Z definiert wie oben; $S_r(x): \mathbb{R}^n \rightarrow \mathbb{R}^1$ mit $S_r(x)=0$ für $x \in Z$;

$S_r(x) > 0$ für $x \notin Z$ und $\forall r \geq 0$ und $\mu_r > 0$ definiert man:

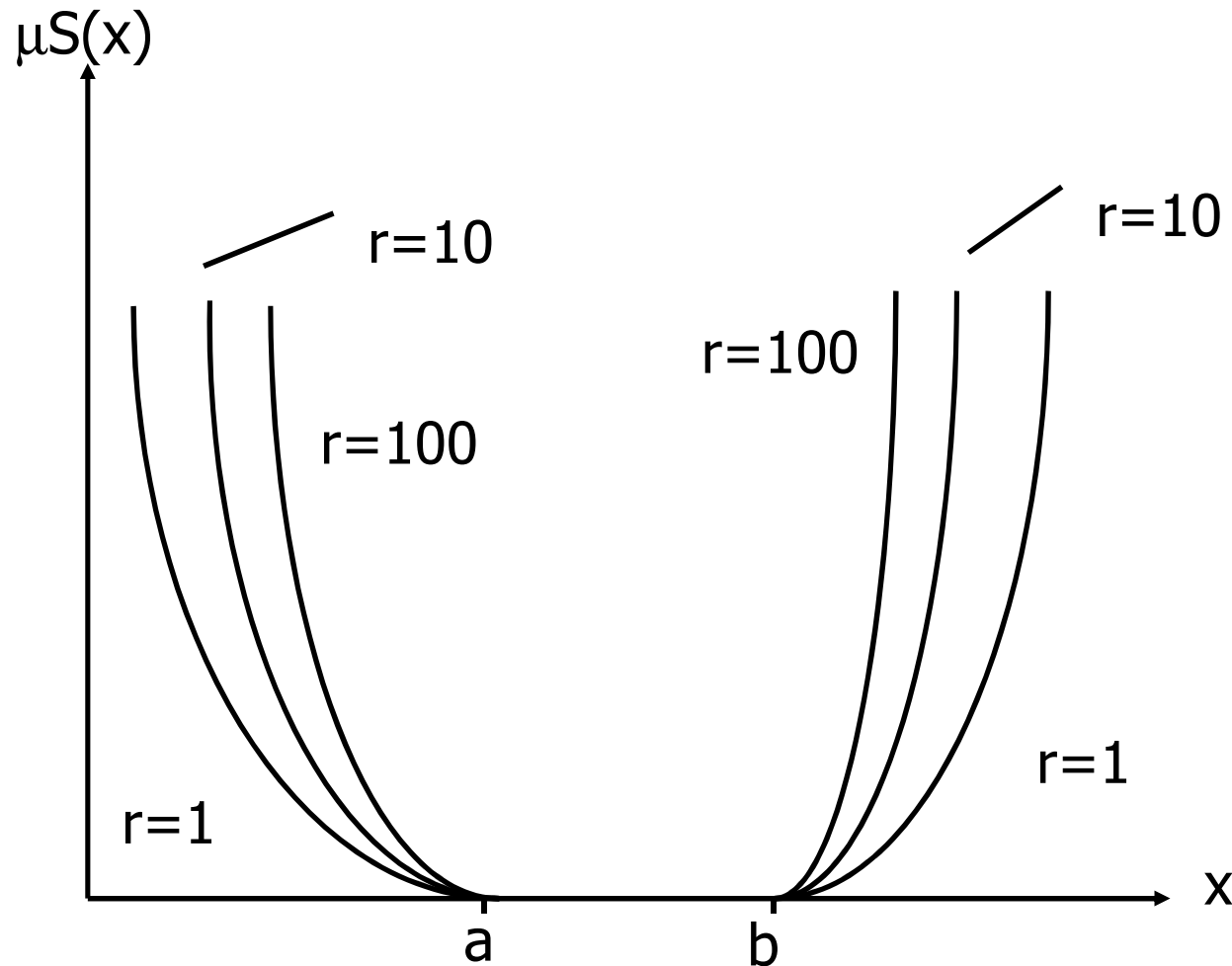
Eine Folge von Funktionen heißt eine Folge von Strafkostenfunktion für den Bereich Z , falls $\forall r, r=0,1,\dots$ gilt:

1. $\mu_r \cdot S_r(x)$ ist stetig in \mathbb{R}^n
2. $\mu_r \cdot S_r(x) = 0 \quad \forall x \in Z$
3. $\mu_{r+1} \cdot S_{r+1}(x) > \mu_r \cdot S_r(x) > 0 \quad \forall r, \forall x \notin Z$ (d.h. monoton wachsend)
4. $\mu_r \cdot S_r(x) \rightarrow \infty \quad \text{für } r \rightarrow \infty \quad \text{und } x \notin Z$

Beispiel: $\min f(x)$ so, dass $g_i(x) \leq 0, i=1,\dots,m, S(x) = \sum_{i=1}^m \max[0, g_i(x)]^2$

Folge: $\min \{f(x) + \mu_r S_r(x)\}_{r=0,1,\dots}$ mit $\mu_r \rightarrow \infty$ für $r \rightarrow \infty$

4.6. Ausgewählte Numerische Verfahren



4.6. Ausgewählte Numerische Verfahren

Verwendung in einem Ansatz (Hilfsproblem)

$\min_{x \in \mathbb{R}^n} [f(x) + \mu \cdot S(x)]$ freies (unrestringiertes) Problem.

Lösung mit entsprechenden Methoden des vorangegangenen Abschnitts.

Parameter $\mu > 0$:

- μ klein geringes Gewicht der Restriktionen; Problem ähnlich
- μ groß Restriktionen dominierend, Verletzung wird stark bestraft.

\Rightarrow Vorgehen: Mit „kleinem“ μ beginnen, dann μ vergrößern, um sich Z zu nähern.

Problemfolge $\{\min_{x \in \mathbb{R}^n} [f(x) + \mu_r \cdot S_r(x)]\}_{r=0,1,\dots}$ lösen.

Algorithmus:

Wähle $\varepsilon > 0$, $\mu_0 > 0$, $\beta > 1$, $x^0 \in \mathbb{R}^n$

Setze $r = 0$

DOWHILE $\mu_r S(x^r) > \varepsilon$

$x^{r+1} := \operatorname{argmin}[f(x) + \mu_r S(x)]$

$\mu_{r+1} := \beta \mu_r$

$r := r + 1$

ENDDO

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Lagrange Relaxation bereits in Abschnitt 1.1 betrachtet.

Wir betrachten jetzt das Mixed Integer Problem (MIP)

$$\text{maximiere} \quad z_{\text{MIP}} = c^T x$$

$$\text{so dass:} \quad A^1 x \leq b^1 \quad (m_1 \text{ „schwierige“ Restriktionen})$$

$$A^2 x \leq b^2 \quad (m_2 \text{ „einfache“ Restriktionen})$$

$$x \in R^n \cap X$$

Durch die Wahl von X , wird erreicht dass das Optimierungsproblem ein MIP ist.

Zu gegebenem $\pi \in R_+^{m_1}$ sind die optimalen Werte der EV \bar{x}

$$\bar{x} = x(\pi) \in X \cap \{x \in R^n \mid A^2 x \leq b^2\}$$

des Lagrange – relaxierten Problems zu ermitteln.

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Mit \bar{x} erhält man die obere Schranke

$$z_{LR}(\pi) = c^T \bar{x} + \pi^T \cdot (b^1 - A^1 \bar{x})$$

Dabei ist $\bar{x} \in X^2 := X \cap \{\bar{x} \in R^n \mid A^2 \bar{x} \leq b^2\}$

Die Idee beim Lagrange – dualen Problem ist, die Werte der Lagrange – Multiplikatoren $\pi \in R_+^{m_1}$ so zu bestimmen, dass das Lagrange-relaxierte Problem eine beste (d.h. kleinste) obere Schranke liefert.

$$z_{LD} = \min_{\pi \geq 0} z_{LR}(\pi) = \min_{\pi \geq 0} \{c^T \bar{x} + \pi^T \cdot (b^1 - A^1 \bar{x})\}$$

Verfahren: zahlreiche gute Verfahren, hier: Subgradientenoptimierung (auch allgemein für konkave bzw. konvexe Optimierungsprobleme)

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

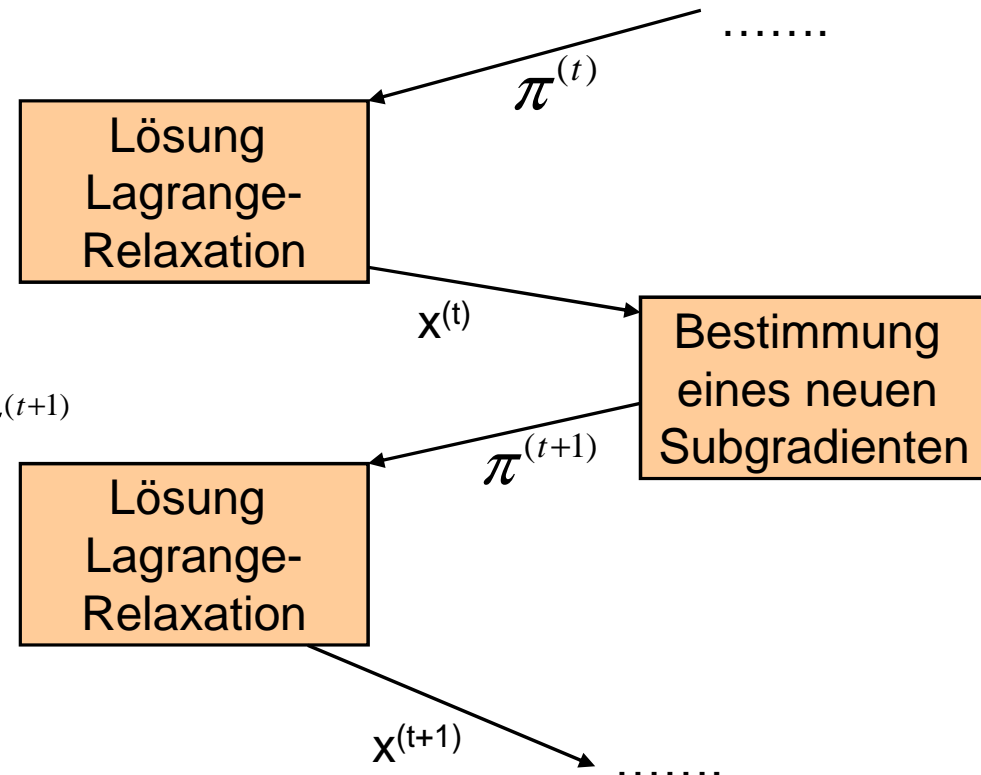
Subgradientenverfahren zur Lösung des Lagrange-dualen Problems

Iterative Wiederholung von

1. Bestimme eine optimale Lösung $x^{(t)}$ der Lagrange-Relaxation für einen gegebenen Wert $\pi^{(t)}$ der Lagrangemultiplikatoren.
2. Bestimme einen nächsten Wert $\pi^{(t+1)}$ aufgrund der opt. Lösung der LR $x^{(t)}$ und des sich daraus ergebenden Subgradienten $s^{(t)}$.

$t \in \{0, 1, \dots\}$

Iterationszähler



4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Subgradienten für konvexe (konkave) Funktionen

$z: R^{m_1} \rightarrow R$ konvexe (bzw. konkave) Funktion

Konvexität: $z(\alpha\pi^1 + (1-\alpha)\pi^2) \leq \alpha \cdot z(\pi^1) + (1-\alpha)z(\pi^2)$

für alle $\pi^1, \pi^2 \in R^{m_1}$ und $0 \leq \alpha \leq 1$.

Diese Definition der Konvexität ist äquivalent mit:

Für jedes $\bar{\pi} \in R^{m_1}$ existiert ein Vektor,

$s = s(\bar{\pi}) \in R^{m_1}$ so dass

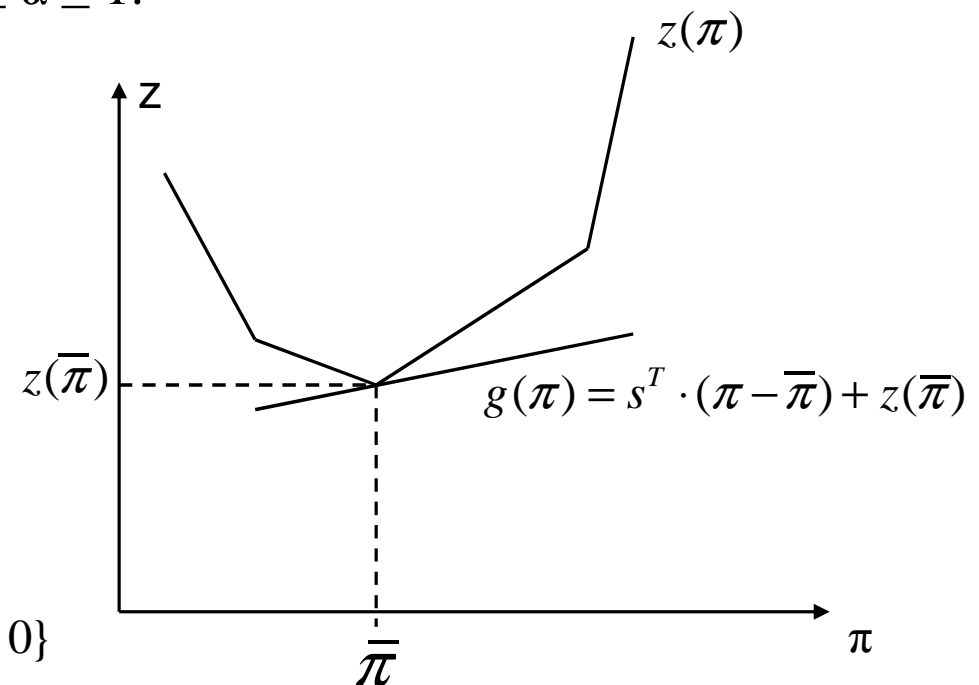
$$z(\bar{\pi}) + s^T \cdot (\pi - \bar{\pi}) \leq z(\pi)$$

für alle $\pi \in R^{m_1}$.

anschaulich: Die durch $s(\bar{\pi})$ definierte stützende Hyperebene

$$\{(\pi, z) \in R^{m_1+1} \mid s^T \cdot (\pi - \bar{\pi}) = 0\}$$

im Punkt $(\bar{\pi}, z(\bar{\pi}))$ verläuft komplett “unterhalb“ des Graphen von $z: R^{m_1} \rightarrow R$



4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Im Bild:

Konvexe, stückweise lineare Funktion. (an den Knickstellen nicht differenzierbar).

Skalar s^T (im allgemeinen ein Vektor s^T) und die dadurch definierte stützende Gerade (i.a. stützende Hyperebene)

$$z = g(\pi) = s^T \cdot (\pi - \bar{\pi}) + z(\bar{\pi}).$$

Die durch s definierte lineare Funktion würde bei einer differenzierbaren Funktion mit der Tangente in $(\bar{\pi}, z(\bar{\pi}))$ an die Funktion z übereinstimmen. (bei konkaven Funktionen analoge Vorgehensweise)

Definition: Es sei $z: R^{m_1} \rightarrow R^1$ eine konvexe Funktion. Dann ist $s \in R^{m_1}$ ein Subgradient an der Stelle $\bar{\pi}$, falls

$$z(\bar{\pi}) + s^T \cdot (\pi - \bar{\pi}) \leq z(\pi) \quad \text{für alle } \pi \in R^{m_1} \text{ gilt.}$$

Gradienten sind an einer differenzierbaren Stelle eindeutig, Subgradienten dagegen an einer nicht-differenzierbaren Stelle nicht eindeutig bestimmt.

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Beispiel – Rucksackproblem

Lagrange-Relaxation:

$$\begin{aligned} \text{maximiere } z_{\text{LR}}(\pi) &= \sum_{j=1}^n (p_j - \pi \cdot w_j) x_j + \pi \cdot C \\ \text{so dass } x_j &\in \{0,1\} \text{ für alle } j = 1, 2, \dots, n \end{aligned}$$

Der Zulässigkeitsbereich $S = \{0,1\}^n$ enthält $K=2^n$ Punkte x^k

$$x^k \in \{0,1\}^n, k = 1, \dots, K$$

Jedem dieser Punkte entspricht ein Zielfunktionswert der Lagrange-Relaxation

$$z_{x^k}(\pi) = \sum_{j=1}^n p_j \cdot x_j^k + \pi \cdot (C - \sum_{j=1}^n w_j x_j^k)$$

Für gegebenes x_j^k ist $z_{x^k}(\pi)$ linear in π .

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Für ein festes π ist derjenige Punkt x^k von Interesse der $z_{x^k}(\pi)$ maximiert.

$$z_{LR}(\pi) = \max_{k=1,\dots,K} z_{x^k}(\pi)$$

d.h. Maximum einer endlichen Zahl linearer Funktionen $\rightarrow z_{LR}(\pi)$ ist als Funktion von π konvex.

Die Knickstellen sind die Schnittpunkte linearer Funktionen

$$z_{LR}(\pi, x^k) \text{ und } z_{LR}(\pi, x^l) \quad \text{für } k \neq l$$

Zahlenbeispiel: $\max. z = 4x_1 + 7x_2 + 5x_3$

$$\text{so dass } 4x_1 + 5x_2 + 3x_3 \leq 10$$

$$x_j \in \{0,1\} \text{ für } j = 1,2,3$$

$$\{0,1\}^3 = \{0,1\} \times \{0,1\} \times \{0,1\}$$

enthält $2^3 = 8$ Punkte

$$x^1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, x^2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, x^3 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, x^4 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, x^5 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, x^6 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, x^7 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, x^8 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Lagrange – Relaxation: maximiere

$$z_{LR}(\pi) = (4 - 4\pi)x_1 + (7 - 5\pi)x_2 + (5 - 3\pi)x_3 + 10\pi,$$

so dass $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in \{x^k \mid k = 1, \dots, 8\}$

Das ergibt die Zielfunktionswerte (für jedes π)

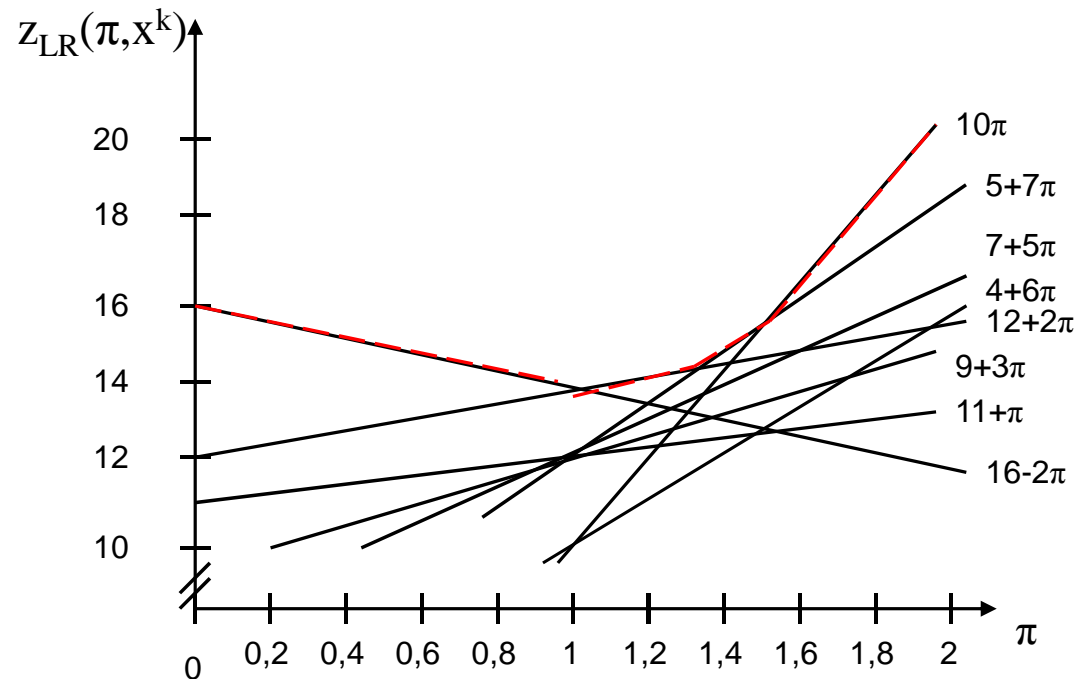
k	1	2	3	4	5	6	7	8
$z_{LR}(\pi, x^k)$	10π	$4+6\pi$	$7+5\pi$	$11+\pi$	$5+7\pi$	$9+3\pi$	$12+2\pi$	$16-2\pi$

Für variables π , $0 \leq \pi \leq 2$, erhält man durch Maximumbildung (punktweise Maximierung) die konvexe Funktion $z_{LR}(\pi)$

$$\min_{\pi} z_{LR}(\pi) \quad , \text{ wird angenommen für } \pi = 1, \quad z_{LR}(1) = 14$$

Der Optimalwert des Rucksackproblems ist $z = 12$. Damit erzeugt die LR eine duale Lücke von $14 - 12 = 2$.

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung



Zielfunktionswerte der Funktionen $z_{LR}(\pi, x^k)$ für verschiedene Extrempunkte x^k in Abhängigkeit von π .

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Zurück zum allgemeinen Fall

Man kann auch allgemein zeigen, dass $z_{LR}(\pi)$ (Zielfunktion des Lagrange-dualen Problems eines MIP (wie oben definiert)) konvex in π ist. $z_{LR}(\pi)$ ist das Maximum einer endlichen Menge linearer Funktionen, also stückweise linear und an den Knickstellen nicht differenzierbar.

Wie bestimmt man das Minimum einer solchen Funktion bzgl. π ?

- Iteratives Verfahren (ähnlich wie bei differenzierbaren Funktionen), das
zunächst an einer Stelle $\bar{\pi}$ einen Subgradienten $s(\bar{\pi})$ bestimmt (den Gradienten $\nabla z_{LR}(\bar{\pi})$ berechnet),
danach eine Schrittweite $w^{(t)} > 0$ in Richtung des negativen Subgradienten geht und das Verfahren dort wiederholt
(Ist $s = 0$ ein Subgradient an der Stelle $\bar{\pi}$, dann ist $\bar{\pi}$ optimal.)

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Algorithmus – Subgradientenoptimierung

- 1: (Initialisierung)** Bestimme einen Startpunkt π . Setze $\pi^{(1)} := \pi$ sowie $t:=1$
- 2: (Subgradienten bestimmen)** Berechne einen Subgradienten $s^{(t)} = s^{(t)}(\pi^{(t)})$
- 3: (Abbruchkriterium)** Falls $s^t = 0$ ist oder falls 0 ein zulässiger Subgradient ist, stop.
- 4: (Schrittweitenbestimmung)** Berechne eine Schrittweite $w^{(t)} \in \mathbb{R}$.
- 5: (Update)** Setze $\pi^{(t+1)} = \pi^{(t)} - w^{(t)} \cdot s^{(t)}$
Falls eine Komponente $\pi_i^{(t+1)} < 0$, setze $\pi_i^{(t+1)} := 0$
Erhöhe Iterationszähler $t := t+1$ und gehe zu 2.

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Bemerkungen:

- Die Überprüfung im Schritt 3 ob 0 ein zulässiger Subgradient ist, ist praktisch schwer realisierbar. → Andere Terminierungskriterien werden benutzt.

(Feste Anzahl von Iterationen, Erreichen einer vorgegebenen Schranke für den ZF-Wert $z_{LR}(\pi)$.)

- Schrittweitenbestimmung ist anders als bei den Gradientenverfahren:

Die Konvergenz des Subgradientenverfahrens gegen ein Optimum lässt sich für eine Schrittweitenfolge $(w^{(t)})_{t \in \mathbb{N}}$, die mit t gegen 0 geht, beweisen. Die Summe der Schrittweiten muss dabei gegen unendlich gehen:

$$w^{(t)} \rightarrow 0 \text{ für } t \rightarrow \infty \quad \underline{\text{und}} \quad \sum_{t=1}^N w^{(t)} \rightarrow \infty \quad \text{für } N \rightarrow \infty$$

- Beispiel: geometrische Folge $w^{(t)} := w^{(0)} \cdot \rho^t$ mit $0 < \rho < 1$.

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Methode von Polyak

Große Schritte durchführen, wenn man weit vom Optimum entfernt ist. In der Nähe des Optimums kleine Schritte gehen.

Idee

z_{MIP} Optimum des Ausgangsproblems, \bar{z} untere Schranke für dieses Optimum.

$\rho^{(t)}$ Parameter mit: $\rho^{(1)}$ (z.B. = 2) wird festgelegt und nach je T Iterationen (z.B. T = 100) halbiert.

=> Schrittweitenformel von Polyak:

$$w^{(t)} := \frac{\rho^{(t)} \cdot (z_{LR}(\pi^{(t)}) - \bar{z})}{\|s^{(t)}\|^2}$$

4.7. Lagrange Dualität für MIP und Subgradientenoptimierung

Berechnung eines Subgradienten für Lagrange – duales Problem

Es sei $\bar{x} = \bar{x}(\bar{\pi})$ die Lösung des Lagrange – relaxierten Problems $LR(\bar{\pi})$ für $\bar{\pi} \in R^{m_1}$

Dann ist $s = b^1 - A^1 \bar{x}$

ein Subgradient der Funktion $z_{LR}(\pi)$ an der Stelle $\bar{\pi}$.

Man zeigt dies durch Einsetzen des Subgradienten in die Definitionsrestriktion und unter Verwendung der Konvexität von $z_{LR}(\pi)$.

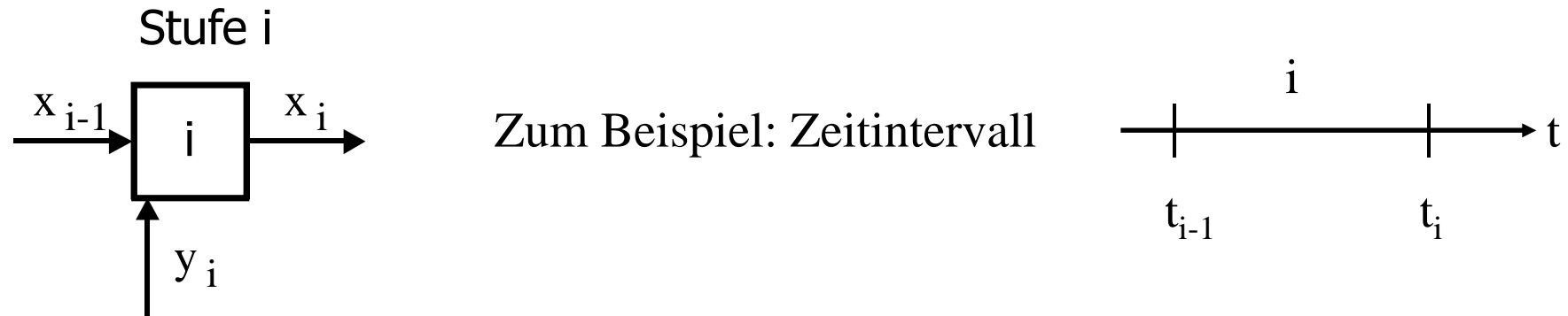
- Da die Lagrange-Relaxation nur für $\pi \geq 0$ definiert ist, setzt man beim Update die Multiplikatoren:

$$\pi_i^{(t+1)} := \max\{\pi_i^{(t)} - w^{(t)} s_i^{(t)}, 0\}$$

5. Dynamische Optimierung und Lagerhaltung

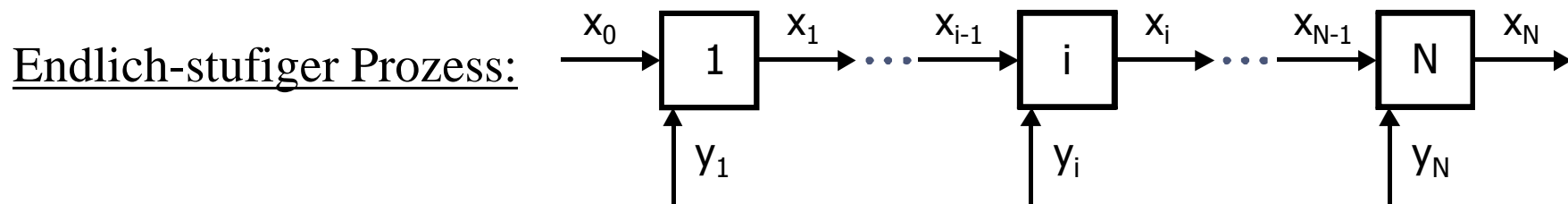
(R. Bellman 1955-1957)

5.1. Mehrstufige Entscheidungsprozesse



x_i – Zustand (am Ende) der Stufe i (Ausgang Stufe i \rightarrow Eingang Stufe i+1)

y_i – Entscheidung auf Stufe i



5.1. Mehrstufige Entscheidungsprozesse

N – Stufen ; alternativ: unendlich viele Stufen

Basisvariante: $x_i \in \mathbb{R}^1$, $y_i \in \mathbb{R}^1$

x_0 – Anfangszustand , x_N – Endzustand

Stufenbewertung: $c_i(x_{i-1}, y_i)$, reellwertige Funktion, „Kosten“

Zustandstransformation / Zustandsgleichung:

$x_i = T_i(x_{i-1}, y_i)$ für alle $i = 1, 2, \dots, N$

T_i gegebene (in Basisvariante reellwertige) Funktion

Entscheidungspolitik (Strategie): $y = (y_1, y_2, \dots, y_N)$

Zustandstrajektorie (von x_0 ausgehend): $x = (x_0, x_1, \dots, x_N)$

N – stufiger Entscheidungsprozess: (y, x) (von x_0 ausgehend)

5.1. Mehrstufige Entscheidungsprozesse

Weitere Restriktionen:

$y_i \in Y_i \subseteq \mathbb{R}^1$, Y_i – Entscheidungsbereich

$x_i \in X_i \subseteq \mathbb{R}^1$, X_i – Zustandsbereich

Zielkriterium:

$Z(x,y) = F(c_1(x_0,y_1), c_2(x_1,y_2), \dots, c_N(x_{N-1}, y_N), c_{N+1}(x_N))$ sei o.E.d.A. zu minimieren

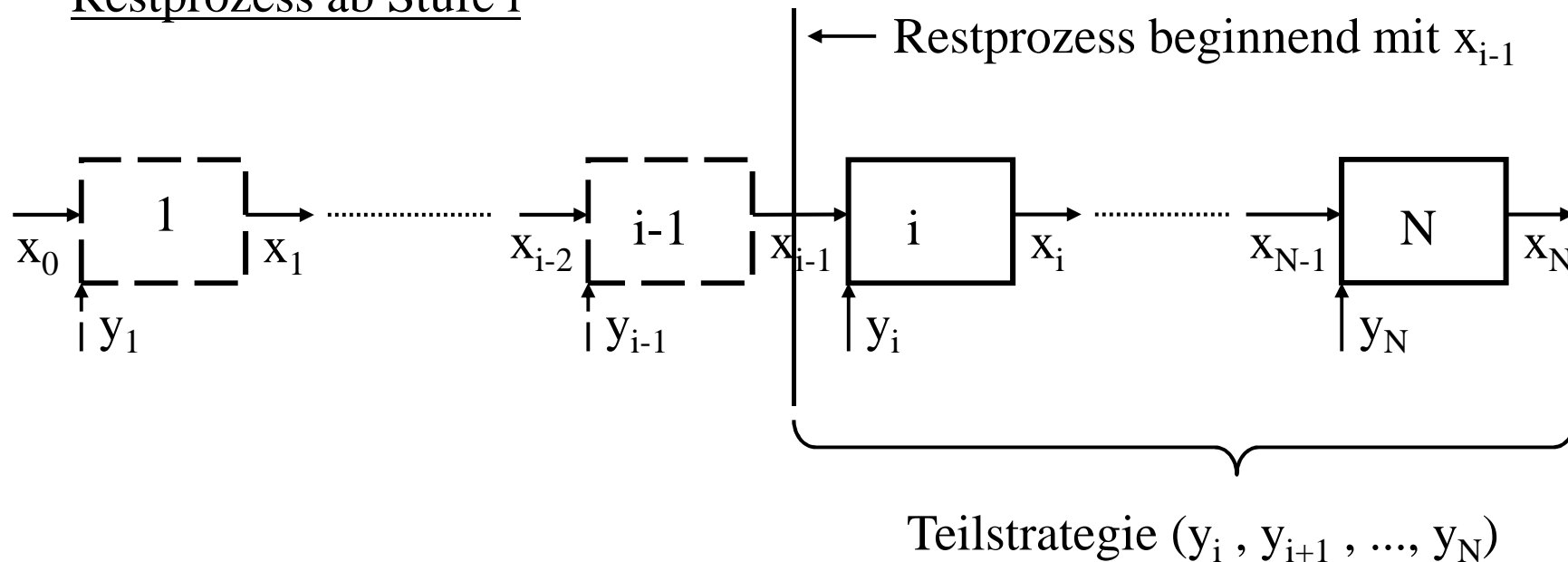
Optimierungsproblem:

Für gegebenen Anfangszustand $x_0 \in X_0$ wähle man eine Entscheidungspolitik y mit $y_i \in Y_i$ so, dass für den N -stufigen Entscheidungsprozess gilt:

$x_i = T_i(x_{i-1}, y_i)$ und $x_i \in X_i$ für alle $i = 1, 2, \dots, N$ und dass die Zielfunktion $Z(x,y)$ ihren minimalen Funktionswert annimmt.

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

Restprozess ab Stufe i



beginnt mit Stufe i im Zustand x_{i-1} . (Für $i=1$ erhält man den Gesamtprozeß)

Idee: Zerlegung des Optimierungsproblems mit $N - \text{Variablen}$ in N
Optimierungsprobleme mit je einer Variablen

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

Bellman'sches Optimalitätsprinzip:

Für jedes $i = 1, \dots, N$ gilt: Die Optimalität des Restprozesses ab Stufe i mit der Strategie $(y_i, y_{i+1}, \dots, y_N)$ hängt nur vom Anfangszustand x_{i-1} dieses Restprozesses ab. Sie hängt nicht davon ab, durch welche Entscheidungen y_1, y_2, \dots, y_{i-1} dieser Zustand x_{i-1} entstanden ist.

Illustration: $N = 3$; $F = \sum$

$$x_0, x_1 = T_1(x_0, y_1), \quad x_2 = T_2(x_1, y_2), \quad x_3 = T_3(x_2, y_3)$$
$$Z(y_1, y_2, y_3) = c_1(x_0, y_1) + c_2(x_1, y_2) + c_3(x_2, y_3) + c_4(x_3)$$

(Es sei $c_4(x_3) = 0$.)

$$\text{Min}_{y_1, y_2, y_3} Z(y_1, y_2, y_3) = \text{Min}_{y_1} \{ c_1(x_0, y_1) + \text{Min}_{y_2} (c_2(x_1, y_2) + \text{Min}_{y_3} c_3(x_2, y_3)) \}$$

(Existenz aller Minima vorausgesetzt. Begründung: Unabhängigkeit!)

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

$$f_N(x_{N-1}, y_N) = f_3(x_2, y_3) := c_3(x_2, y_3)$$

$$f_3^*(x_2) = \min_{y_3} f_3(x_2, y_3) = \min_{y_3} c_3(x_2, y_3)$$

$$\rightarrow f_2^*(x_1) = \min_{y_2} \{ c_2(x_1, y_2) + f_3^*(x_2) \} \quad \text{mit } x_2 = T_2(x_1, y_2)$$

usw.

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

Prinzip!

Für welche Problemklassen kann man zeigen, dass das Prinzip erfüllt (also eigentlich ein mathematischer Satz) ist?

Allgemein gilt:

Jede additive Zielfunktion (ohne zusätzliche Nebenbedingungen) der Art

$$Z(x,y) = \sum_{i=1}^N c_i(x_{i-1}, y_i) \quad \text{erfüllt das Prinzip.}$$

Es ergibt sich eine Folge zusammenhängender Optimierungsprobleme:

$$f_{N+1}^*(x_N) := 0$$

$$f_i(x_{i-1}, y_i) := c_i(x_{i-1}, y_i) + f_{i+1}^*(T_i(x_{i-1}, y_i))$$

$$f_i^*(x_{i-1}) := \min_{y_i} f_i(x_{i-1}, y_i), \text{ für } i = N, \dots, 1$$

$$\hat{y}_i = \hat{h}_i(x_{i-1})$$

d.h. N Teilprobleme (parametrische Optimierungsprobleme)

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

$$\min_{y_i} f_i(x_{i-1}, y_i), \text{ für } i = N, \dots, 1$$

← parametrische Probleme

Dann gilt:

Ist $y_i \in Y_i$ (Y_i darf von x_{i-1} abhängen), $Y_i = Y_i(x_{i-1})$, dann liefert jede Strategie, die sich als Lösung der oben definierten Folge von Optimierungsproblemen ergibt, mit Sicherheit optimale Prozesse (abhängig von x_0) und $f_1^*(x_0)$ ist das Minimum der Zielfunktion $f_1^*(x_0) = \min_y Z(x_0; x_1, \dots, x_N; y_1, \dots, y_N)$

Wie liefert die Folge der Opt. Probleme eine Strategie?

$i=N$

$$f_N(x_{N-1}, y_N) = c_N(x_{N-1}, y_N) + 0$$

$$f_N^*(x_{N-1}) = \min_{y_N \in Y_N(x_{N-1})} f_N(x_{N-1}, y_N)$$

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

$Y_i(x_{i-1})$ entstehen z.B., wenn Zustandsnebenbedingungen $x_i \in X_i$ gefordert werden.

Falls das obige Minimierungsproblem für jedes feste $x_{N-1} = x_{N-1}^\circ$ eindeutig lösbar ist, so existiert eine Funktion \hat{h}_N mit:

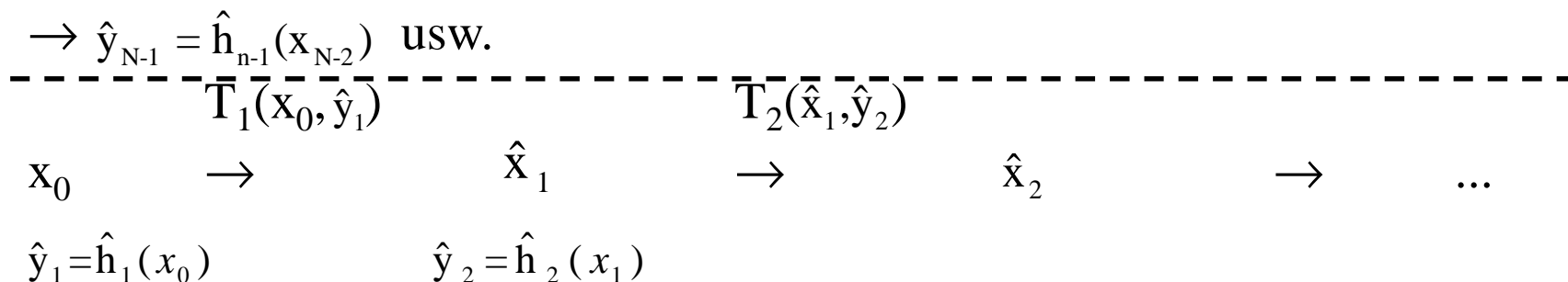
$\hat{y}_N = \hat{h}_N(x_{N-1})$ optimale Entscheidungsfunktion für Stufe N (bei bekanntem x_{N-1} !)

$$f_{N-1}(x_{N-2}, y_{N-1}) = c_{N-1}(x_{N-2}, y_{N-1}) + f_N^*(x_{N-1})$$

mit $x_{N-1} = T_{N-1}(x_{N-2}, y_{N-1})$

$$f_{N-1}^*(x_{N-2}) = \min_{y_{N-1} \in Y_{N-1}(x_{N-2})} f_{N-1}(x_{N-2}, y_{N-1})$$

$$\rightarrow \hat{y}_{N-1} = \hat{h}_{N-1}(x_{N-2}) \text{ usw.}$$



5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

Allgemeinere Probleme bzw. Modifikationen

1. Terminales Zielkriterium

Jedes Problem mit einer Zielfunktion

$$z(x, y) = \sum_{i=1}^N c_i(x_{i-1}, y_i) + c_{N+1}(x_N) \quad (*)$$

kann man auf ein Problem mit $z(x, y) = \varphi_{N+1}(x_N)$ zurückführen
(Preis: eine zusätzliche Zustandsgröße und Zustandsgleichung pro Stufe).

Modell: $x_i = T_i(x_{i-1}, y_i)$, $i=1, \dots, N$ und $(*)$

$$z_1 = c_1(x_0, y_1) \quad (= \varphi_1(x_0, y_1))$$

$$z_2 = z_1 + c_2(x_1, y_2) \quad (= \varphi_2(x_1, z_1, y_2)) \dots$$

$$z_i = z_{i-1} + c_i(x_{i-1}, y_i) \quad (= \varphi_i(x_{i-1}, z_{i-1}, y_i))$$

$$i = 1, 2, \dots, N \quad \text{mit } z_0 = 0, \quad z_{N+1} = z_N + c_{N+1}(x_N)$$

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

es gilt:

$$z_{N+1} = z(x, y) = \varphi_{N+1}(x_N, z_N) = z_N + c_{N+1}(x_N)$$

2. Vektoren statt Skalare im Modell und in den FG

$$x_i \in \mathbb{R}^n, \quad i = 0, \dots, N$$

$$y_j \in \mathbb{R}^m, \quad j = 1, \dots, N$$

Vektortransformation

$$x_i = T_i(x_{i-1}, y_i), \quad i=1, \dots, N \quad \text{z.B.: } x_i = Ax_{i-1} + By_i + c_i$$

$$y_j \in Y_j(x_{j-1}) \subseteq \mathbb{R}^m \text{ Zustandsabhängige Entscheidungsmenge, } j=1, \dots, N,$$

$$\text{z.B. } Cy_j + Dx_{j-1} = d_j$$

$$z = z(x, y) = \varphi_{N+1}(x_N) \Rightarrow \min$$

FG sind formal identisch.

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

3. Nicht-additive Zielfunktionen – monotone Separabilität

Definition:

$Z(x, y)$ heißt separabel, falls gilt:

\exists Verknüpfungsoperatoren “o” und Zerlegungsfunktionen g_i , so dass z darstellbar ist als: $z(x, y) = g_1(c_1(x_0, y_1), \dots, c_N(x_{N-1}, y_N))$

mit $g_N(c_N(x_{N-1}, y_N)) := c_N(x_{N-1}, y_N)$

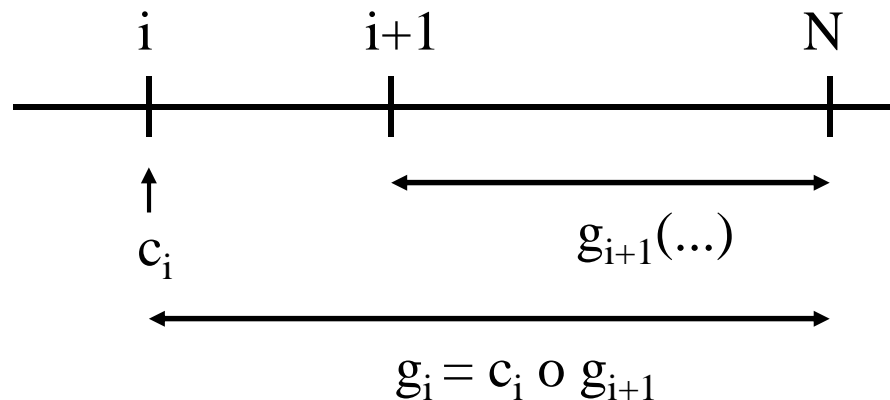
$g_i(c_i(x_{i-1}, y_i), \dots, c_N(x_{N-1}, y_N)) := c_i(x_{i-1}, y_i) \text{ o } g_{i+1}(c_{i+1}(x_i, y_{i+1}), \dots, c_N(x_{N-1}, y_N))$

Operator o abhängig von i

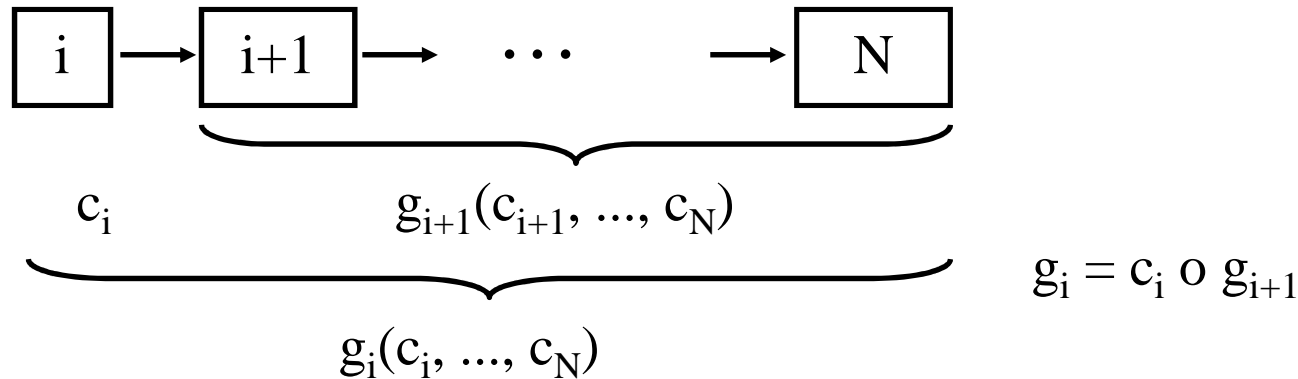
Beispiele:

- alle o sind +
- alle o sind \cdot , min, max
- $c_1 + c_2 \cdot c_3$

$(c_1 + c_2) \cdot (c_3 + c_4)$ nicht separabel!



5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen



Hilfssatz

Falls $z(x, y)$ separabel ist und für alle i gilt:

Die Funktionen g_i , $g_i = c_i \circ g_{i+1}$ sind für jeden zulässigen Wert von c_i monoton wachsend in g_{i+1} , dann folgt:

$$\begin{aligned}
 \rightarrow \min_{y_i, \dots, y_N} g_i(c_i, g_{i+1}) &= \min_{y_i} g_i(c_i, \min_{y_{i+1}, \dots, y_N} g_{i+1}) \\
 &= \min_{y_i} (c_i \circ \min_{y_{i+1}, \dots, y_N} g_{i+1})
 \end{aligned}$$

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

Beweis: Nemhauser, G.L.

Einführung in die Praxis der Dynamischen Optimierung,
München – Wien, 1969

→ Rekursionsformeln (Dynam. Opt.) gelten deshalb (Hilfssatz) analog
zum additiven Fall, vorausgesetzt es gilt:

monotone Separabilität

Rekursionsgleichungen

Dyn. Opt. bei monoton separablen Problemen

$$f_{N+1}^*(x_N) := 0 \text{ bzw. } 1 \text{ (oder } \phi_{N+1}(x_N))$$
$$f_i(x_{i-1}, y_i) := c_i(x_{i-1}, y_i) \circ f_{i+1}^*(T(x_{i-1}, y_i))$$

$$f_i^*(x_{i-1}) := \min_{y_i} f_i(x_{i-1}, y_i), \quad (\text{bzw. max})$$
$$(\text{ = } \min_{y_1, \dots, y_N} g_i(c_i, \dots, c_N)), \quad (\text{bzw. max})$$

$$i = N, \dots, 1$$

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

$$f_1^*(x_0) = \min_{y_1, \dots, y_N} z, \quad (\text{bzw. max})$$

$$f_i^*(x_{i-1}) = \min_{y_i} [c_i(x_{i-1}, y_i) \circ f_{i+1}^*(x_i)]$$

Marketing-Beispiel

Die Unternehmung H. hat für die Einführung eines neuen Waschmittels einen Etat von 5 Mio. Euro. Die Produkteinführung soll in drei Phasen abgewickelt werden:

Phase 1: Herabgesetzte Einführungspreise, um Erstkäufer zu gewinnen.

Phase 2: Werbekampagne: Erstkäufer überzeugen, dass der reguläre Preis akzeptabel ist.

Phase 3: Sonderangebote und Werbemaßnahmen, um die regulären Käufer zu halten.

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

Man verteile den Etat optimal auf die drei Phasen.

→ Man muß die Ausgaben in den Phasen mit resultierendem Marktanteil in Verbindung bringen.

Schätzungen auf der Basis von Marktanalysen

Ausgaben in Phase 1 [Mio Euro]	0	1	2	3	4	5
Resultierender Marktanteil [in %]	0	10	15	22	27	30

Ausgaben in Phase 2 [Mio Euro]	0	1	2	3	4
Anteil gehaltener Kunden [in %]	30	55	70	80	85

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

Ausgaben in Phase 3 [Mio Euro]	0	1	2	3	4
Anteil gehaltener Kunden [in %]	50	70	85	90	93

M_1 : in der Phase 1 erzielter Marktanteil $M_1(y_1)$

M_2, M_3 : davon in Phase 2 (3) gehaltener Marktanteil $M_2(y_2), M_3(y_3)$

y_i : in Phase i eingesetzter Betrag (in Mio. Euro), $i=1, 2, 3$

Offenbar muß man $y_1 \geq 1$ fordern!

Modell

max $M_1 \circ M_2 \circ M_3 \cdot 1 \quad (M_4(x_3) = 1)$

mit $y_1 + y_2 + y_3 \leq 5, \quad y_1 \geq 1, \quad y_2, y_3 \geq 0$ und ganzzahlig

Normierung der M_i auf $0 \leq M_i \leq 1$ durch Division durch 100%.

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

Dynamische Optimierung

Stufen $i = 1, 2, 3$ (entsprechend den Phasen)

EV: y_i

Zustände: Rest-Etats nach Stufen i

$x_0 = 5$ (5 Mio. sollen eingesetzt werden)

$x_i = x_{i-1} - y_i$ und $x_3 = 0$

Rekursion:

$$f_4^*(x_3) = 1$$

$$f_i^*(x_{i-1}) = \max [M_i(y_i) \circ f_{i+1}^*(x_{i-1} - y_i)], \quad i = 3, 2, 1$$

Da alle $M_i \geq 0$ sind, ist die Zielfunktion monoton separabel.

$i = 3$

$$x_3 = 0 \rightarrow x_3 = x_2 - y_3 = 0; y_3 = x_2$$

$$f_3^*(x_2) = \max [M_3(y_3) \cdot 1] = M_3(x_2) \text{ und } y_3^* = h_3^*(x_2) = x_2$$

(Die Endbedingung erzwingt die Entscheidung der letzten Stufe.)

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

x_2	0	1	2	3	4	
$y_3^* = h_3^*(x_2)$	0	1	2	3	4	$= x_2$
$f_3^*(x_2)$.50	.70	.85	.90	.93	$= M_3(x_2)$

$$\underline{i = 2}$$

$$x_2 = x_1 - y_2 \geq 0 \Leftrightarrow y_2 \leq x_1$$

$$f_2^*(x_1) = \max_{0 \leq y_2 \leq x_1} \underbrace{[M_2(y_2) \circ f_3^*(x_1 - y_2)]}_{= f_2(x_1, y_2)}$$

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

x_1	$f_2(x_1, y_2)$					y_2^*	$f_2^*(x_1)$
	$y_2=0$	$y_2=1$	$y_2=2$	$y_2=3$	$y_2=4$		
0	.150	-	-	-	-	0	.150
1	.210	.275	-	-	-	1	.275
2	.255	.385	.350	-	-	1	.385
3	.270	.468	.490	.400	-	2	.490
4	.279	.495	.595	.560	.425	2	.595

$$f_2(3, 0) = M_2(0) \circ f_3^*(3-0) = 0.30 \cdot 0.90 = 0.270$$

$$f_2(4, 3) = M_2(3) \circ f_3^*(4-3) = 0.80 \cdot 0.70 = 0.56$$

5.2. Bellmansches Optimalitätsprinzip und Funktionalgleichungen

$$\underline{i=1} \quad f_1^*(5) = \max_{1 \leq y_1 \leq x_0=5} [M_1(y_1) \circ f_2^*(5-y_1)]$$

	$y_1=1$	$y_1=2$	$y_1=3$	$y_1=4$	$y_1=5$	y_1^*	$f_1^*(5)$
$x_0=5$.0595	.0735	.0847	.0743	.0450	3	.0847

$$f_1(5, y_1) = M_1(y_1) \circ f_2^*(5-y_1)$$

Optimale Politik:

In Phase 1 3 Mio., danach je 1 Mio. einsetzen.

Optimaler Marktanteil: 8,47 %

5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)

Dynamisches deterministisches Lagerhaltungsmodell

Planungszeitraum endlich $[t_a, t_e]$

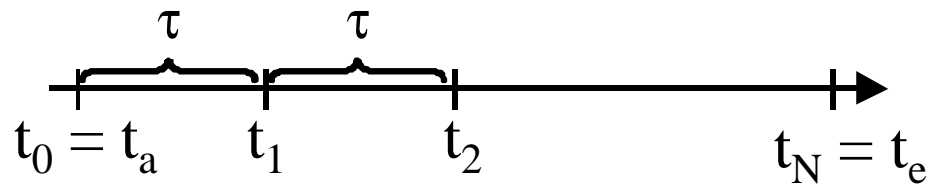
Nachfragerate über $[t_a, t_e]$ nicht konstant

$[t_a, t_e]$ in N gleich lange Perioden eingeteilt

$t_0 := t_a, t_1, \dots, t_{N-1}, t_N := t_e$ Endpunkte der Perioden

$\tau := t_j - t_{j-1}$ für $j=1, \dots, N$

5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)



Voraussetzungen

λ - Lieferzeit: Bestellzeitpunkte $t'_j := t_j - \lambda$, $j = 0, \dots, N-1$
(\rightarrow Bestellung zu den Zeitpunkten t_j verfügbar)

$z_j > 0$ Nachfrage in der j -ten Periode $[t_{j-1}, t_j]$, $j=1, \dots, N$

$u_j \geq 0$ zu Beginn von Periode j gelieferte Bestellmenge

x_j - Lagerbestand am Ende der Periode j ($t_j - \epsilon$)
(unmittelbar vor Auffüllen zur Zeit t_j), $j=1, \dots, N$

Lagerbilanzgleichung

$$x_j = x_{j-1} + u_j - z_j, \quad j = 1, \dots, N \quad (1)$$

Fehlmengen sind nicht zugelassen: $x_j \geq 0$, $j = 0, \dots, N$

Anfangs- und Endlagerbestand vorgegeben: o.E.d.A.: $x_0 = x_N = 0$

5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)

Nachfrageentwicklung in j-ter Periode:

$\beta_j(t)$: der bis zur Zeit t in der Periode j (gerechnet vom Beginn der Periode) angefallene Bedarf, $0 \leq t \leq \tau$ mit $\beta_j(0) = 0$, $\beta_j(\tau) = z_j$

Kosten:

K: fixe Bestellkosten (für alle Perioden gleich)

c: Preis pro ME

h: Lagerungskosten pro ZE und ME

Lagerungskosten in der j-ten Periode

$$K \cdot \delta(u_j) + c \cdot u_j + h \cdot \int_0^{\tau} (\underbrace{x_{j-1} + u_j}_{=x_j + z_j} - \beta_j(t)) dt =$$

$$K \cdot \delta(u_j) + c \cdot u_j + h \cdot \tau \cdot x_j + h \cdot \int_0^{\tau} (z_j - \beta_j(t)) dt$$

Diskussion von $h \cdot \int_0^{\tau} (z_j - \beta_j(t)) dt$:

5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)

“Kosten für die Mindestreserve”

(Lagerungskosten für die zu Beginn der j-ten Periode zur Verfügung gestellten z_j ME, um Nachfrage in dieser Periode zu befriedigen)

$$\xi_j = \frac{1}{\tau} \cdot \int_0^{\tau} (z_j - \beta_j(t)) dt \quad \text{“mittlere Mindestreserve” in Periode } j$$

→ Kosten für Mindestreserve: $h \cdot \tau \cdot \xi_j$

Konstante Nachfragerate: $\beta_j(t) = \frac{z_j}{\tau} \cdot t \rightarrow \xi_j = \frac{z_j}{2}$, d.h. bisheriges Modell
Kosten für Mindestreserve sind unabhängig von Bestellmenge!

Zur Optimierung der Bestellpolitik genügt es, die Minimierung von

$$\sum_{j=1}^N [K \cdot \delta(u_j) + \underbrace{c \cdot u_j}_{=c \cdot (z_j + x_j - x_{j-1})} + h \cdot \tau \cdot x_j] \quad (2) \quad \text{zu betrachten.}$$

Es gilt:

$$\begin{aligned}
 \sum_{j=1}^N u_j &= \sum_{j=1}^N (x_j - x_{j-1} + z_j) \\
 &= \sum_{j=1}^N z_j + x_N - x_0 \\
 &= \sum_{j=1}^N z_j, \text{ also konstant.}
 \end{aligned}$$

Dynamisches Optimierungsmodell

$$\min \sum_{j=1}^N \underbrace{[K \cdot \delta(u_j) + h \cdot \tau \cdot x_j]}_{f(u_j, x_j)} \quad (3)$$

$$\text{so dass } x_j = x_{j-1} + u_j - z_j, \quad j = 1, \dots, N \quad (1)$$

$$x_0 = x_N = 0 \quad (4)$$

$$\left. \begin{aligned} x_j &\geq 0, \quad j = 1, \dots, N-1 \\ u_j &\geq 0, \quad j = 1, \dots, N \end{aligned} \right\} (5)$$

5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)

Lösungsansatz:

In (3) x_j anstelle $x_{j-1} \wedge (1)$ ist eindeutig nach x_{j-1} auflösbar
→ Optimierungsrichtung der Dynamischen Optimierung
(Rückwärtsrechnung) umkehrbar

Bellmansche Rekursionsformeln:

$$C_0^*(x_0) = 0 \quad (6)$$

$$C_j^*(x_j) = \min_{0 \leq u_j \leq x_j + z_j} [K \cdot \delta(u_j) + h \cdot \tau \cdot x_j + C_{j-1}^*(x_{j-1})]$$

mit $x_{j-1} = x_j - u_j + z_j$, für $j=1, \dots, N$

und es gilt: $C_N^*(x_N = 0)$: Minimum der relevanten Kosten

Lösung: Numerische Lösung von (6) oder zunächst Ausnutzung der speziellen Struktur von (6) zur Aufwandsreduzierung.

5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)

Eigenschaften der Bellman-FG:

(u_1^*, \dots, u_N^*) optimale Bestellpolitik,

$(x_0^*, x_1^*, \dots, x_N^*)$ zugehörige Folge (gemäß (1))

$x_j = x_{j-1} + u_j - z_j$ von Lagerbeständen mit $x_0^* = x_N^* = 0$.

Dann heißt (x_{j-1}^*, u_j^*) ein “optimales Lösungspaar”, d.h.
aktueller Bestand \rightarrow nachfolgende Bestellung

(E1) Für ein optimales Lösungspaar (x_{j-1}^*, u_j^*) gilt entweder

$$x_{j-1}^* = 0 \wedge u_j^* \geq 0 \text{ oder } x_{j-1}^* > 0 \wedge u_j^* = 0, \quad j = 1, \dots, N$$

(E2) Eine optimale Bestellung u_j^* in der j-ten Periode kann nur einen der $N-j+2$ Werte $0, z_j, z_j+z_{j+1}, \dots, z_j+z_{j+1}+\dots+z_N, j=1, \dots, N$ annehmen.

5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)

Damit läßt sich die Bellman-FG (6) vereinfachen:

$$C_0^* = 0, \quad C_1^* = K, \quad C_j^* = \min_{k=1, \dots, j} \{p_j(k)\}$$

$$\text{mit} \quad p_j(k) := K + h\tau \cdot \sum_{i=k}^j \sum_{\gamma=i+1}^j z\gamma + C_{k-1}^*$$

gilt:

$$C_1^* = p_1(1) = K$$

$$p_j(\kappa_j^*) = \min_{k=1, \dots, j} p_j(k) = C_j^*$$

Wird das Minimum für mehr als einen Index k angenommen, sei κ_j^* irgendeiner dieser Indizes (z.B. der größte Index).

5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)

Algorithmus

- Für $j = 1$ ist $\kappa_j^* = \kappa_1^* = 1$
- Obige FG in Vorwärtsrechnung für $j = 2, \dots, N$ lösen, d.h. C_j^* und κ_j^* ermitteln
- Optimale Bestellmengen durch Rückwärtsrechnung bestimmen:
sei $s := \kappa_N^* \rightarrow u_s^* = z_s + z_{s+1} + \dots + z_N \wedge u_{s+1}^* = \dots = u_N^* = 0$
ist weiter $m := \kappa_{s-1}^* \rightarrow u_m^* = z_m + z_{m+1} + \dots + z_{s-1} \wedge u_{m+1}^* = \dots = u_{s-1}^* = 0$

Die Rechnung kann durch Ausnutzung folgender Eigenschaften stark vereinfacht werden:

- (E3) Für die κ_j^* gilt: $\kappa_1^* \leq \kappa_2^* \leq \dots \leq \kappa_N^*$
 \rightarrow Bei Lösung der FG nur Minimum der Werte $p_j(\kappa_{j-1}^*), \dots, p_j(j)$ bilden. (Auf Betrachtung von $p_j(1), \dots, p_j(\kappa_{j-1}^* - 1)$ kann man verzichten.)

5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)

(E4) Die $p_j(k)$ genügen für $j = 2, \dots, N$ der Rekursion

$$p_j(k) = p_{j-1}(k) + (j-k) \cdot h \cdot \tau \cdot z_j, \quad k = 1, \dots, j-1$$

$p_j(j) = K + C_{j-1}^* \rightarrow$ Die explizite Berechnung der Doppelsumme kann entfallen.

Minimale gesamte Lagerungskosten C^* :

$$C^* = C_N^* + c \cdot \sum_{j=1}^N z_j + h \cdot \tau \cdot \sum_{j=1}^N \xi_j'$$

Ein Zahlenbeispiel

$$K = 40 \quad [\text{Euro}]$$

$$c = 0,5 \quad [\text{Euro/ME}]$$

$$h = 0,2 \quad [\text{Euro/ME} \cdot \text{Monat}]$$

$$\tau = 1 \quad [\text{Monat}]$$

5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)

j	z _j [ME]
1	200
2	100
3	400
4	300

Konstante Nachfrage in den Perioden, mittlere Mindestreserve in der j-ten Periode

$$\xi_j = \frac{z_j}{2}, \quad j=1, 2, 3, 4$$

Vorwärtsrechnung:

1) $C_1^* = p_1(1) = K = 40, \quad \kappa_1^* = 1$ Startwerte ~ Initialisierung

2) $j = 2: p_2(1) = p_1(1) + h \cdot \tau \cdot z_2 = 60$

$$p_2(2) = K + C_1^* = 80$$

→ Minimum → $C_2^* = 60, \kappa_2^* = 1$

5.3. Anwendungen in der Lagerhaltung (Wagner-Whitin Methode)

$$\begin{aligned} 3) \quad j = 3: p_3(1) &= p_2(1) + 2 \cdot h \cdot \tau \cdot z_3 = 220 \\ p_3(2) &= p_2(2) + h \cdot \tau \cdot z_3 = 160 \\ p_3(3) &= K + C_2^* = 100 \quad \leftarrow \min \\ \rightarrow C_3^* &= 100, \kappa_3^* = 3 \end{aligned}$$

$$\begin{aligned} 4) \quad j = 4: p_4(3) &= p_3(3) + h \cdot \tau \cdot z_4 = 160 \\ p_4(4) &= K + C_3^* = 140 \quad \leftarrow \min \\ \rightarrow C_4^* &= 140, \kappa_4^* = 4 \end{aligned}$$

Rückwärtsrechnung:

$$\begin{aligned} s_1 &:= \kappa_4^* = 4 \rightarrow u_4^* = z_4 = 300 \text{ [ME]} \\ s_2 &:= \kappa_{s_1-1}^* = \kappa_3^* = 3 \rightarrow u_3^* = z_3 = 400 \text{ [ME]} \\ s_3 &:= \kappa_{s_2-1}^* = \kappa_2^* = 1 \rightarrow u_1^* = z_1 + z_2 = 300 \text{ [ME]} \\ u_2^* &= 0 \end{aligned}$$

$$C^* = C_4^* + c \cdot \sum_{j=1}^4 z_j + h \cdot \tau \cdot \sum_{j=1}^4 \xi_j = 740 \text{ [Euro]}$$