Was muss man wissen?
Operations Research II

Patrick Elftmann und Matthias Sondermann
studium@sp3cialman.net

24. Oktober 2005

Vorbemerkung

Dies ist eine kleine Zusammenfassung des prüfungsrelevanten Stoffes der Vorlesung 'Operations Research II' bei Prof. Sebastian. Wir haben einfach mal versucht, die wichtigsten Sätze auszuformulieren und die wichtigsten Beweisideen zusammenzufassen. Dabei weisen wir ausdrücklich daraufhin, dass es definitiv Lücken gibt, so dass man dieses Werk hier definitiv als unvollständig und falsch titulieren kann. Aber alles andere war nie unser Anspruch. Es soll einfach nur einen kleinen Überblick verschaffen mehr nicht.

Es erhebt keinerlei Anspruch auf Korrektheit und Vollständigkeit und stellt keine offizielle Veröffentlichung des Lehrstuhls dar. Bei Fehlern, Verbesserungsvorschlägen oder sonstigen Anregungen wird um eine Email an die angegebene Adresse gebeten.

Inhaltsverzeichnis

1 Zuordnungsproblem **3**
- 1.1 Das Modell . . . 3
- 1.2 Die Ungarische Methode . . . 3

2 Das Transport- und Umladeproblem **5**
- 2.1 Das Transportproblem . . . 5
- 2.2 Das Umladeproblem . . . 9
- 2.3 Beziehungen zwischen Zuordnungs-, Transport- und Umladeproblem . . . 10

3 Das Rucksackproblem **11**
- 3.1 Das Modell . . . 12
- 3.2 1. Branch&Bound-Algorithmus . . . 12
- 3.3 2. Branch&Bound Algorithmus . . . 12

4 Tourenplanung **13**
- 4.1 Das Traveling Salesman Problem - TSP . . . 13
  - 4.1.1 Nearest-Neighbour-Heuristik . . . 13
- 4.2 Das Vehicle Routing Problem - VRP . . . 15
  - 4.2.1 Savings-Methode (Methode von Clarke and Wright) . . . 17
  - 4.2.2 Sweep-Verfahren (Methode von Gillet und Miller) . . . 18
  - 4.2.3 Verbesserungsverfahren für TSP und VRP . . . 19

5 Branch&Bound **19**

6 Gemischt-ganzzahlige Programmierung **21**
- 6.1 Ansatz von Dakin . . . 21

7 Lagerhaltungsmodelle **22**
7.1 Klassisches Losgrößenmodell 22
7.2 Klassisches Losgrößenmodell mit Fehlmengen 24
7.3 Dynamische Optimierung in der Lagerhaltung 25
7.4 Verfahren von Wagner und Whitin 26

8 Dynamische Optimierung **28**

9 Nichtlineare Optimierung **31**
9.1 Konvexe Optimierung und Kuhn-Tucker-Theorie 31
9.2 Quadratisches Programmieren 35
9.2.1 Algorithmus von Wolfe 36
9.3 Verfahren der Nichtlinearen Programmierung 36
9.3.1 Mehrdimensionales Newtonverfahren 37
9.3.2 Verfahren zulässiger Richtungen 38
9.3.3 Minimierung mit Nebenbedingungen 39

1 Zuordnungsproblem

verbale Formulierung

In der Kostenmatrix sind n Elemente so auszuzeichnen, dass in jeder Zeile und in jeder Spalte genau eine Auszeichnung besteht. Die Summe der Bewertungen der ausgezeichneten Elemente sei möglichst klein.

1.1 Das Modell

- Instanz: m × n Bewertungsmatrix $C = (c_{ij})_{mn}$ mit m=n

Modell

Gegeben Bewertungsmatrix C

Zielfunktion: $z = \sum_{i,j} c_{ij} \cdot x_{ij} \rightarrow \min$

Zeilenbedingung: $\sum_{j=1}^{n} x_{ij} = 1$ für jedes i = 1, …, m

Spaltenbedingung: $\sum_{i=1}^{m} x_{ij} = 1$ für jedes j = 1, …, n

Entscheidungsvariablen: $x_{ij} = \begin{cases} 1 & \text{, Element (i,j) wird ausgezeichnet} \\ 0 & \text{, sonst} \end{cases}$

Greedy-Heuristik

Man zeichne nacheinander das kleinste jeweils noch mögliche Elemente aus

1.2 Die Ungarische Methode

- **Satz:**
  Die optimale Zuordnung ändert sich nicht, wenn zu einer Zeile oder Spalte der Bewertungsmatrix C eine Konstante addiert wird.

  ↷ Die optimale Zuordnung ändert sich nicht, wenn

  - in jeder Zeile das Zeilenminimum **(Zeilenreduktion)** und anschließend
  - in jeder Spalte das Spaltenminimum **(Spaltenreduktion)** subtrahiert wird.

- Ergebnis ist eine Matrix mit mindestens einer Null in jeder Spalte und jeder Zeile.

- Gelingt eine vollständige Zuordnung, bei der nur Nullen ausgezeichnet sind, ist die gefundene Lösung offenbar eine optimale Lösung.
  Gelingt dies nicht, so ist das nächstbeste Element auszuzeichnen (heuristischer Ausweg) $\curvearrowright$ Lösung ist nicht zwingend optimal

- Ausweg: Mit Hilfe von 'Alternierende Wege' andere Nullen auszeichnen

- **Prozedur ALTERNIERENDE WEGE:**

```
DOWHILE   Es gibt einen alternierenden Weg, der mit einer
          Null in einer Zeile ohne Auszeichnung beginnt und bei
          einer freien Null endet.
          Vertausche die Rollen von freien und ausgzeichneten
          Nullen auf diesem Pfad.
ENDDO
```

- Gelingt dies auch nicht, so ist zu jeder Zeile und Spalte von C eine Konstante derart zu addieren, dass die Matrix nicht-negativ bleibt, aber mindestens eine neue Null ensteht.

- Lassen sich in einer n $\times$ n Bewertungsmatrix (auch nach Ausführen der Prozedur "Alternierende Wege") nur K $< n$ Nullen auszeichnen, so existieren K Linien durch Zeilen und Spalten, die alle Nullen überdecken.

- **Prozedur ÜBERDECKUNG:**

```
Streiche alle Spalten mit einer ausgezeichneten Null,
die auf einem Sackgassen-Weg liegen.
Streiche alle Zeilen, in denen eine ausgezeichnete
Null noch nicht durchgestrichen ist.
```

  Diese Prozedur erzeugt genau K $<$ n Linien und es ist jede Null durchgestrichen.

- **Regel:**
  Subtrahiere m von allen nicht überdeckten Elementen und addiere m zu allen Kreuzungspunkten.
  Dabei sei m der minimale Wert der nicht überdeckten Elemente.

  Hintergrund:
  Zu allen gestrichenen Zeilen und Spalten m/2 addieren und von allen nicht gestrichenen Zeilen und Spalten m/2 subtrahieren.

- Es ensteht ein Problem, für das die gleiche Zuordnung optimal ist und das mindestens eine neue Null besitzt.

- Nach voller Zuordnung neu suchen, d.h. alternierenden Pfad suchen.

- Erstauszeichnung möglichst geschickt vornehmen:
  - zuerst nur die Zeilen und Spalten mit nur einer Null
  - dann jeweils möglichst eine Zeile oder Spalte auszeichnen, für die eine Auszeichnung zwingend ist

Algorithmus UNGARISCHE METHODE

```
1. Zeilen- und Spaltenreduktion
2. Möglichst weitgehende Auszeichnung von Nullen
3. CALL ALTERNIERENDE WEGE
   DOWHILE noch keine volle Zuordnung
      CALL ÜBERDECKUNG
      m:= Minimum aller nicht gestrichenen Zahlen
      Subtrahiere m von allen nicht gestrichenen Zahlen
      Addiere m zu allen Kreuzungspunkten
      Nimm eine möglichst weitgehende Auszeichnung von Nullen vor
      CALL ALTERNIERENDE WEGE
   ENDDO
```

Der Algorithmus bricht nach endlich vielen Schritten mit einer Optimallösung ab.

2 Das Transport- und Umladeproblem

2.1 Das Transportproblem

verbale Formulierung

Ein homogenes Gut, das an den Orten A_i (i=1,...,m) in den Mengen a_i angeboten wird, soll zu den Orten B_j (j=1,...,n) transportiert werden, an denen eine Nachfrage von b_j besteht. Die Kosten des Transportes einer Mengeneinheit (ME) von Ort A_i zum Ort B_j betragen c_{ij}. Dabei gilt es zu bestimmen, wie viele ME von Ort A_i zum Ort B_j transportiert werden sollen, so dass die Nachfrage erfüllt wird und die Gesamttransportkosten minimal sind.

Alternativ:
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.

Das Transportproblem entspricht dem Umladeproblem ohne Umladeorte.

Modellierung

- Vorraussetzung: (homogenes) Gut, Erzeugnis
- Problem:

1. **m Ausgangsorte** A_i, i = 1, ..., m,
   in denen ein Erzeugnis zur Verfügung steht (Lager, Produzenten)
   a_i - Mengeneinheiten (Aufkommen)
2. **n Bestimmungsorte** B_j, j = 1, ..., n
   die einen Bedarf an diesem Erzeugnis haben, (Kunden, Produzenten, Lager)
   b_j - Mengeneinheiten (Bedarf)
3. Belieferung direkt von jeweils einem A_i zu einen B_j.
4. c_{ij}: Transportkosten pro Mengeneinheit (m · n Relationen)
5. **Ausgeglichenheit des Problems:**
   Der Gesamtbedarf ist gleich dem Gesamtaufkommen: $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$

- **Modell:**
  Transportierte Menge des Erzeugnisses von A_i nach B_j sei x_{ij},
  i = 1, 2, ..., m; j = 1, 2, ..., n (in Mengeneinheiten des Erzeugnisses)

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

  unter $\quad \sum_{j=1}^{n} x_{ij} = a_i$ für i = 1, 2, ..., m

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

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

  falls $\quad \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$, so ist das Problem ausgeglichen.

Heuristiken

1. **Greedy-Heuristik:**
   minmales Element maximal belegen
2. **NordWestEcken-Regel**:
   - Beginne in der Nordwestecke (NWE) und setze $x_{11} = \min(a_1, b_1)$
   - 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')
   - Iteriere bis alle x_{mn} festgelegt sind.
     Die entstehende Lösung ist eine ZBL für das Transportmodell.

3. **VAM-Heuristik** (Vogelsche Approximations 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 Oppertunitätskosten. Sie sind gleich der Differenz (D_i bzw. D_j) zwischen den zweitniedrigensten 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.

   (c) Im Feld (i,j) setze $x_{ij} = \min(a_i, b_i)$, 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 Spalte (in der übrig gebliebenen Zeile oder Spalte entsteht eine BV = 0)

   (d) Überprüfe, ob alle Zeilen- oder Spaltenbedingungen erfüllt sind.
   Wenn ja "STOPP", sonst gehe zu a)

Die Modi (Modified Distribution)- Methode

- $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$ 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			...		...			...		a_1
	2			...		1	1	...	1	...			...		a_2
	...			...				...		...			...		...
	m			...				...		...	1	1	...	1	a_m
Zeile	1	1		...		1		...		...	1		...		b_1
	2		1	...			1	...		...		1	...		b_2
	...			...				...		...			...		...
	n			...	1			...	1	...			...	1	b_n

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

Lösung mit der Simplexmethode

Tableau hätte m+n Zeilen, m · n Spalten; Basis (m+n-1)×(m+n-1) Teilmatrix von Rang m+n-1, da die m+n Zeilen nicht linear unabhängig sind. Man verifiziere, dass die Differenz aus der Summe der ersten m und der Summe der letzten n Zeilen eine

Nullzeile ergeben würde. Also gilt:
Wegen Ausgeglichenheit des Transportproblems genügen **m+n-1 Basisvariablen**.

- Spezielle Struktur der Koeffizientenmatrix A (nur 0 und 1, speziell angeordnet) erlaubt folgende Schlußfolgerungen:
  - B^{-1} enthält ausschließlich +1, -1, 0
  - $B^{-1}b = x_B$ ganzzahlig, falls b ganzzahlig
  - Vereinfachtes Tableau möglich, da die Strukturen von A, B, B^{-1} bekannt sind.
  - Es genügt, in jeder Iteration die Basis, die Basislösung und die Δz_j der NBV zu bestimmen.
- zulässige Ausgangsbasislösung z.B. mittels NWE-Regel
- Grundidee: **Verwendung des dualen Modells**

  maximiere $\quad Z = \sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j$

  unter $\quad u_i + v_j \leq c_{ij}$ für i = 1, ..., m; j = 1, ..., n

- Die dualen Strukturvariablen sind wegen der Gleichungsrestriktionen des primalen Modells unbeschränkt.
- Das Gleichungssystem des primalen Modells ist unterbestimmt $\curvearrowright$
  Einer dualen Variable 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 Basisvariablen 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 Gleichungsproblem für alle Basisvariablen.
- Für die aufzunehmende BV braucht man Δz_{ij}. Diese ergeben sich hier zu
  $\Delta z_{ij} = c_{ij} - (u_i + v_j)$
- **Optimalitätsbedingung:**
  Die Lösung ist optimal, wenn für alle NBV $\Delta z_{ij} \geq 0$ gilt.
- **Aufnahmebedingung:**
  Es ist die NBV x_{rs} in die Basis aufzunehmen, für die $\Delta z_{rs} = \min_{i,j} \Delta z_{ij} < 0$ gilt.
- **Basistausch - zu elimierende Basisvariable:**
  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.
   Sobald bei Erhöhung der NBV eine mit Minus markierte ursprüngliche Basisvariable einen Wert < 0 annimmt, würde Unzulässigkeit eintreten. Deshalb: Ausscheidende Basisvariable x_{kl}^B gemäß $\theta = \min_{i,j}\{x_{ij}^B\} = x_{kl}^B$ (mit Minus markiert) bestimmen.
   θ gibt auch den Wert der aufzunehmenden NBV an (Pivotisierungsprozess).

2.2 Das Umladeproblem

verbale Formulierung

Beim Umladeproblem ist der kostengünstigste Transport eines Gutes von gewissen Angebotsorten (Anbietern oder Produzenten eines Gutes) über sogenannte Umladeort (z.B. Zwischenlager oder Monatgeorte) zu Nachfrageorten (Verbrauchern des Gutes) gesucht.

Beispielsweise kann es sich um den kostengünstigsten Transport von Wasser in einem Wasserversorgungsnetz handeln (die Kosten können etwa durch den Energiebedarf für das Pumpen des Wassers sowie durch Wassverluste u.a. bedingt sein). Die Angebotsorte entsprechen hierbei Wasserwerken, die Wasser in das Netz einspeisen, die Nachfrageorte Verbrauchern (Gemeinden) und die Umaldeorte Pumpstationen oder Verzweigungspunkten des Netzes.

- **Entscheidungsvariablen:**
  x_{ij} auf dem Pfeil (i,j) transportierte Menge (Interpretation: Fluß)
- **Restriktionen:**
  Jeder Pfeil besitzt eine Maximalkapazität κ_{ij} zur Aufnahme des Mengenflusses ($\kappa_{ij} = \infty$ zugelassen), $0 \leq x_{ij} \leq \kappa_{ij}$ für alle (i,j) $\in$ E.
- **Flussbedingung:**
  $$\underbrace{\sum_{j \in S(i)} x_{ij}}_{Nachfolgeknoten} - \underbrace{\sum_{k \in P(i)} x_{ki}}_{Vorgängerknoten} = a_i \text{ für i = 1, 2, ..., n}$$
- **Kosten** für den Transport auf (i,j): c_{ij}
- **Modell:**

Minimiere $\sum_{(i,j)\in E} c_{ij} \cdot x_{ij}$

unter $\sum_{j\in S(i)} x_{ij} - \sum_{k\in P(i)} x_{ki} = a_i$ für i = 1,2, ..., n

$0 \leq x_{ij} \leq \kappa_{ij}$ für alle (i,j) $\in$ E

o.B.d.A $\sum_{i=1}^{n} a_i = 0$

(ansonsten Einführung eines fiktiven Nachfrage- bzw. Angebotsortes)

- **Lösung:**
  Das Umladeproblem ist ein spezielles LP mit oberen Grenzen für die Variablen $\curvearrowright$ Netzwerk-Simplex-Algorithmus

2.3 Beziehungen zwischen Zuordnungs-, Transport- und Umladeproblem

- Das **Zurordnungsproblem** kann **als spezielles Transportproblem** aufgefasst werden.

  Aus dem Modell des Transportproblems mit $m = n, a_i = 1 \forall i, b_j = 1 \forall j$

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

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

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

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

  folgt das Zuordnungsproblem

  - Das Zuordnungsproblem hat gleiche Anzahl an Anbietern und Nachfragern
  - Die Vorräte und Bedarfe sind je 1

  Daraus folgt, dass Transportproblemalgorithmen auf Zuordnungsprobleme angewendet werden können.

- Jedes **Transportproblem** kann **als Zuordnungsproblem** formuliert werden.
  Idee:

  - Für jeden Vorratsknoten i mit dem Vorrat a_i: a_i Knoten in 1. Farbklasse anlegen
  - Für jeden Bedarfsort j mit Bedarf b_j: b_j Knoten in 2. Farbklasse anlegen

  - Zurodnungskosten für Paar (i,j) gleich den Einheitstransportkosten festlegen
  - Gegenbenenfalls Dummy-Objekte (Knoten) erzeugen

- Das **Transportproblem** kann **als spezielles Umladeproblem** aufgefasst werden
  - Das spezielle Umladeproblem hat keine Umladeorte
  - Das spezielle Umladeproblem besitzt keine unteren oder oberen Kapazitätsgrenzen

- Das **Zuordnungsproblem** ist ein **Spezialfall des Umladeproblems**

$$\text{Minimiere} \quad \sum_{(i,j)\in E} c_{ij} \cdot x_{ij}$$

$$\text{unter} \quad \sum_{j\in S(i)} x_{ij} = 1$$

$$\sum_{i\in P(j)} x_{ij} = 1$$

$$x_{ij} \geq 0$$

  - Die Interpretation des Zuordnungsproblems als Umladeproblem besagt, dass ein (ganzzahliger) kostenminimaler Fluss der Stärke n in dem zugrunde liegenden Netzwerk N von den Quelle i zu den Senken j gesucht ist
  - Dabei sind c_{ij} die Kosten für den Transport einer Mengeneinheit auf der Kante (i,j) $\in$ E
  - Auf allen Kanten sind die Minimalkapazitäten gleich 0 und die Maximalkapazitäten gleich 1

3 Das Rucksackproblem

verbale Formulierung

Dem Rucksackproblem liegt folgende anschauliche Aufgabenstellung zugrunde:
Ein Wanderer kann in seinen Rucksack verschiedene Ausrüstungsgegenstände einpacken, wobei die einzelnen Gegenstände unterschiedliche Gewichte und Nutzen haben. Gesucht ist eine optimale Rucksackfüllung, d.h. eine Füllung mit maxiamalem Gesamtnutzen, wobei ein vorgegebenes Gesamtgewicht des Rucksacks nicht überschritten werden darf.

Beispiele: Machienenbelegungsplan, Frachtladungsprobleme, ...

3.1 Das Modell

Maximiere $\sum c_j x_j$

unter $\sum a_j x_j \leq b$

$x_j \in \{0,1\}$ für j = 1, ..., n

$$x_j = \begin{cases} 1 & \text{, falls j aufgenommen wird} \\ 0 & \text{, sonst} \end{cases}$$

Eine einzige Restriktion, aber x_j nicht kontinuierlich sondern binär $x_j \in \{0,1\}$ für alle j.

3.2 1. Branch&Bound-Algorithmus

- Der Simplexalgorithmus wird nicht benötigt, kann aber sinnvoll angewendet werden wenn das Problem relaxiert wird.
- **Idee:**
  Teilbaum generieren und Knoten so früh wie möglich als "nicht optimal" erkennen $\curvearrowright$ terminieren (d.h. auf Erzeugung ganzer Teilbäume verzichten)
- **Bound:**
  1. noch erreichbarer Gewinn b_i
  2. verfügbares Schlupf der Restriktion f_i
- **Branching:**

  Knoten mit höchster Schranke 1) zuerst entwickeln
- **Terminierung:**
  1. verfügbarer Schlupf ist zu klein für jedes weitere Element
  2. erreichter Gewinn ist niedriger als bereits realisierter Gewinn
     (eine bessere Lösung ist bekannt)

3.3 2. Branch&Bound Algorithmus

- **relaxiertes Modell:**
  Ganzzahligkeit nicht mehr gefordert.

  Maximiere $\sum c_j x_j$

  unter $\sum a_j x_j \leq b$

  $0 \leq x_j \leq 1$

- Variablen nach fallenden Quotienten umordnen

  (ZF-Koeffizient / Koeffiezient in Restriktion)

- In der Reihenfolge dieser Ordnung jede Variable auf den größten zulässigen Wert setzen.

4 Tourenplanung

4.1 Das Traveling Salesman Problem - TSP

verbale Formulierung

Beim TSP sucht man eine optimale Reiseroute von einem Startort über eine Anzahl vorgegebener Orte zurück zum Startort, wobei jeder Ort genau einmal besucht wird.

- **Gegeben:**
  Eine Menge von Punkten (Knoten, Orten, Städten) und die Entfernung zwischen je zwei dieser Punkte.

- **Gesucht:**
  Eine kürzeste Rundreise durch alle Punkte (jeder Punkt wird genau einmal besucht und der Anfangspunkt ist gleich dem Endpunkt der Rundreise).

- Es gibt (n-1)! derartige Rundreisen, Lösungsraum wächst exponentiell mit n, denn

$$n! \approx \left(\frac{n}{e}\right)^n \sqrt{\frac{2\pi}{n}}$$

- TSP ist NP-schwer $\curvearrowright$ Anwendung von Heuristiken bei realistischen Probleminstanzen

4.1.1 Nearest-Neighbour-Heuristik

1. Man betrachte den Ausgangspunkt der Rundreise. Initialisiere Liste der noch nicht besuchten Punkte.

2. Wähle den nächstgelegenen Punkt aus, besuche nächstgelegenen Punkt, aktualisiere Liste der noch nicht besuchten Punkte.

3. Falls Liste der noch nicht besuchten Städte leer ist, gehe zurück zum Ausgangspunkt und berechne Gesamtlänge.

- Der Algorithmus hat einen Aufwand von O(n^2) und man kann Fehlerschranken angeben, mit denen man das Optimum höchstens verfehlt.
- Die Nearest-Neighbor-Heuristik liefert nicht automatisch eine optimale Lösung (Rundreise), da zum Schluß gegebenfalls weite Wege gegangen werden müssen.
- Verbesserung der Lösung durch Streichen von Kanten und Aufnahme neuer Kanten, die die Gesamtlösung verbessern.

Beschreibung als Zurordnungsproblem

Bewertungsmatrix C = (c_{ij}) wird aus Graph G=(V,Q) ermittelt.
(V=Knotenmenge, Q=Kantenmenge)

1. $x_{ij} \in \{0,1\}, \forall i \in V$

   $$x_{ij} = \begin{cases} 1 & \text{, fall von Stadt i direkt nach Stadt j} \\ 0 & \text{, sonst} \end{cases}$$

2. $\sum_j x_{ij} = 1, \forall i \in V$

   von jedem Ort aus gibt es genau einen Nachfolgeort in der Tour

3. $\sum_i x_{ij} = 1, \forall j \in V$

   jeder Ort hat genau einen Vorgängerort in der Tour

4. $\sum_{i \in W} \sum_{j \notin W} x_{ij} \geq 1, \forall W \subset V, W \neq \emptyset \neq V$

   aus jeder echten Teilmenge W und V führt mindestens ein Pfeil heraus - keine Kurzzyklen

5. Zielfunktion: min $\sum_i \sum_j c_{ij} x_{ij}$

Lösung durch Branch&Bound

- **Problem R:**
  Relaxation = Zuordnungsproblem 1-3 und 5.
  Ordne jeder Ecke/Knoten eine Nachfolgerecke zu.
- **Verzweigung:**
  Falls Kurzzyklen entstehen, so spalte in Teilprobleme auf. Dabei wird jeweils mindestens ein Pfeil des Graphen verboten ($c_{ij} = \infty$).
- **Verzweigungsregel:**

  - Wähle zunächt einen möglichst kurzen Zyklus mit z Kanten
    $\curvearrowright$ In keiner Lösung des TSP sind alle z Kanten vorhanden (ansonsten Zyklus). Deshalb verzweige in z Teilprobleme, wobei jeweils einer der Pfeile verboten wird.
  - Man kann mehrere Kanten verbieten (kleinerer Baum), weil:
    Von der ausgewählten Komponente des Graphen muss mindestens ein Pfeil in eine andere Komponente führen (Restriktion 4)

    Fallunterscheidung:

    * Fall 1: Es geht ein Pfeil von der ersten der z Ecken (und evtl. auch von weiteren Ecken) zu einer anderen Komponente des Graphen.
    * Fall i: Es geht ein Pfeil von der i-ten der z Ecken zu einer anderen Komponente des Graphen, aber nicht von den Ecken 1 bis i-1.
      Fall i schließt alle Lösungen des TSP aus, die bereits in einem der Fälle 1 bis i-1 zulässig sind.
  - Definiere zu jedem Fall i (i = 1, 2, ..., z) ein Teilproblem R_i.

- Im Problem R_i können folgende Kanten verboten werden (weil gemäß Definition R_i die eine mögliche Kante schon festgelegt ist):

  - Alle Kanten der i-ten Ecke zu den übrigen z-1 Ecken dieser Komponente.
  - Alle Kanten der ersten i-1 Ecken der Komponente zu irgendeiner anderen Komponente.

4.2 Das Vehicle Routing Problem - VRP

verbale Formulierung

Mit einer Menge von Fahrzeugen sind, ausgehend von einem Depot, n Kunden mit einem Gut zu beliefern. Der Bedarf jedes Kunden sowie alle Fahrzeiten (Kosten) sind bekannt. Es sind Fahrten so durchzuführen, dass

- jede Fahrt am Depot beginnt und endet
- jeder Kunde auf genau einer Fahrt bedient wird
- die Fahrzeugkapazität nicht überschritten wird (Kapazitätsrestriktion)
- eine für jede Fahrt gleiche vorgegebene Maximaldauer nicht überschritten wird (Zeitrestriktion)

Gesucht ist eine Menge von Fahrten mit minimaler Gesamtdauer.

Das Modell

- Graph G = (V,A,C)
- Knotenmenge V = {1, ..., n}:
  - Depots R = {1, ..., r}, r < n
  - Kunden i ∈ V \ R = {r+1, ..., 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
- Ziel ist die Kostenminimierung

Nebenbedingungen:

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

  $$\sum_{i \in V \setminus R} d_i \leq D$$
- Beschränkte Anzahl der Orte auf einer Route
- Zeitbeschräkungen: 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 VRP:

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

4.2.1 Savings-Methode (Methode von Clarke and Wright)

- Vorraussetzung für das Savings-Verfahren ist eine symmetrische Entfernungsmatrix.
- Das Verfahren beginnt mit einer Startlösung, bei der jeder Kunde i für sich eine Tour i bildet (**Pendeltour**).
  Die Anzahl n der Touren entspricht somit der Anzahl n der Kunden.
- Die Gesamtlänge Z des Tourenplans berechnet sich durch

$$Z = 2 \cdot \sum d_{0i},\ i \in V \backslash R$$

  wobei d_{0i} die Entfernung des Knoten i vom Depot 0 und V\R die Menge aller Kunden ist.
- Diese ungünstige Ausgangslösung wird nun sukzessive durch Zusammenschluß von jeweils zwei Touren verbessert, wobei vorhandene Kapazitäts- und Zeitrestriktionen berücksichtigt werden.
- **Saving:**

$$\begin{aligned} \text{s(i,j)} &= Z_p - Z_v \\ &= 2 \cdot d_{0i} + 2 \cdot d_{0j} - (d_{0i} + d_{ij} + d_{j0}) \\ &= d_{0i} + d_{0j} - d_{ij} \end{aligned}$$

- Ablauf des Savings-Verfahren:
  1. Eine Tourenliste mit den n Pendeltouren wird erstellt.
  2. Für je zwei Endkunden (der erste und der letzte Kunde einer Route) i,j werden die $n^2 - n$ Savingswerte s(i,j) berechnet und hinterlegt.
  3. Die berechneten Savingswerte werden absteigend sortiert.
  4. Mit dem größten Savingswert beginnend, werden die positiven Savingswerte s(i,j) in absteigender Reihenfolge überprüft, ob unter Einhaltung der Restriktionen die zu den Endkunden i und j zugehörigen Touren verknüpft werden können.
  5. Können in 4. zwei Touren verknüpft werden, wird dies umgesetzt. In der Tourenliste werden die beiden verknüpften Touren gestrichen und die neue Tour wird in die Tourenliste aufgenommen.
  6. In 4. wird der folgend kleinere Savingswert überprüft.
  7. Ende, wenn alle positiven Savingswerte überprüft sind.

4.2.2 Sweep-Verfahren (Methode von Gillet und Miller)

- Der Sweep-Algorithmus von Gillet und Miller (1974) ist ein koordinatenorientiertes Verfahren, bei dem die Lage der Knoten in Form von kartesischen Koordinaten oder Polarkoordinaten vorgegeben ist.
- Die Standorte der Kunden und des Depots durch Koordinaten (x_i, y_i) sind so gegeben, dass das Depot im Ursprung liegt.
- Die Entfernungen d_{ij} zwischen zwei Standorten werden euklidisch ermittelt.
- Die Kunden werden nun nach aufsteigenden Polarwinkeln sortiert und in dieser Reihenfolge von 1 bis n durchnumeriert.
- Der Sweep-Algorithmus ist ein sequentielles Verfahren und baut seine Touren wie folgt auf:
  - Die erste Tour enthält die Kunden 1,2,...,i_1 mit den kleinsten Polarwinkeln.
  - Die zweite Tour wird durch die Kunden $i_1, i_1 + 1, ..., i_1 + i_2$ nach steigendem Polarwinkel gebildet, usw.
  - Bei der Zusammenfassung von Kunden zu Touren werden in einem Segment so viele Kunden zusammengefasst, bis die Kapazitätsgrenzen C oder TD erreicht sind. Dies geschieht mit dem Lösungsverfahren des Traveling Salesman Problems, indem die Heuristik des nächsten Nachfolgers und das Einfügeverfahren angewendet werden.
  - Die für jede Tour zu bestimmende kürzeste Route kann dann mit einem Verbesserungsverfahren (2-opt- oder 3-opt-Verfahren) berechnet werden.
  - In einem weiteren Schritt kann geprüft werden, ob der Austausch des letzten Kunden einer Tour gegen den nächstgelegenen Kunden der nächsten Tour eine Verbesserung bringt.
- Die Zusammenfassung der Kunden in Segmente hängt davon ab, welcher Kunde als Kunde 1 festgelegt wurde.
- Weitere Tourenpläne werden analog erstellt, indem die Toureneinteilung mit Kunde 2,3,...,n beginnt.
  Damit werden n verschiedene Tourenpläne erzeugt, von denen der beste Plan ausgesucht werden kann.
- Der Sweep-Algorithmus erzielt gute Ergebnisse, wenn das Depot zentral relativ zur Kundenstruktur liegt.

4.2.3 Verbesserungsverfahren für TSP und VRP

Verfahren	Nachbarschaft	Bemerkung	Eigung für VRP
k-opt	k Kantentausche		wenig
Or-opt	Verschieben von 1-3 benachbarten Kanten	Teilmenge von 3-opt	gut
2-opt*	Austausch von Touranfangsstücken zw. 2 Touren	Teilmenge von 2-opt, nur für je 2 Touren	gut
CROSS	Austausch je eines beliebigen Tourenabschnittes zw. 2 Touren	enthält 2-opt* und Or-opt Nachbarschaft	gut
λ-Interchange	Austausch von Knotenteilmengen von max λ Knoten zw. 2 Touren	enthält Or-opt-Nachbarschaft	gut

- **2-opt:**
  In einem Teil der Tour wird die Orientierung umgekehrt
  $\curvearrowright$ für Probleme mit Zeitfenstern wenig geeignet

- **OR-opt:**
  Orientierung wird nicht geändert
  $\curvearrowright$ für Probleme mit Zeitfenstern gut geeignet

- **2-opt***:
  Orientierung von Teiltouren werden nicht geändert.
  Kunden am Touranfang bleiben am Anfang, Kunden am Tourende bleiben am Ende
  $\curvearrowright$ für Probleme mit Zeitfenstern gut geeignet

- **λ-Interchange:**
  λ=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 in Tour 1

5 Branch&Bound

verbale Formulierung

Für schwere kombinatorische Optimierungsprobleme kommen in erster Linie zwei Lösungsprinzipien in Frage: Die sogenannte Branch&Bound-Methode, die im allgemeinem einen exponentiellen Rechenaufwand erfordert und stets eine optimale Lösung liefert (falls überhaupt eine existiert), und heuristische Verfahren, die in der Regel polynomialen Rechenaufwand haben, aber im allgemeinem nur suboptimale Lösungen (d.h. Näherungslösungen) bestimmen. Algorithmen, die stets eine optimale Lösung liefern, nennt man auch

exakte Verfahren.
Branch&Bound ist eine implizite Enumeration, bei der man sukzessive versucht, Teilmengen vom Lösungsraum M zu finden, in denen keine optimale Lösung von Problem P liegt und die folglich ausgesondert werden können. Dies kann mit Hilfe eines sogenannten Suchbaumes geschehen, d.h. eines Wurzelbaumes, dessen Knoten gewissen Teilmengen von M entsprechen. Dabei bezieht sich "Branch" auf das "Verzweigen" des Suchbaums, wodurch man neue Teilmengen von M generiert. "Bound" weist auf die Verwendung unterer und oberer Schranken für die Zielfunktionswerte hin, mit deren Hilfe man "uninteressante" Teilmengen von M eliminieren kann.

- Problem P: maximiere f(x)
- Anstelle **ein** Problem P zu lösen, wird eine Folge von Problemen P_k mit Lösungsbereich X_k konstruiert mit dem Ziel,
  - eine optimale Lösung von P_k zu bestimmen, oder
  - zu zeigen, dass der Optimalwert von P_k nicht besser ist, als der bisher beste Zielfunktionswert, oder
  - zu zeigen, dass der Lösungsraum X_k leer ist
- Der Lösungsaufwand des Verfahrens kann verringert werden durch
  - geschicktes Verzweigen in die Teilprobleme (Branching)
  - Terminierung von Teilbäumen **vor** deren vollständiger Generierung (Bounding)
- Grundelemente des Verfahrens:
  1. **Initialisierung**
     (a) Definition des Ausgangsmodells P_1 (Wurzel im Verzweigungsbaum)
     (b) aktuelle untere Schranke $\underline{z}$ bei Maximierungs- und aktuelle obere Schranke $\overline{z}$ bei Minimierungsproblem angeben
     (zulässige Lösung bekannt oder $\underline{z} = -\infty$ bzw. $\overline{z} = \infty$)
  2. **Verzweigung (Branching)**
     (a) Auswahl eines noch nicht untersuchten Teilmodells oder
     (b) Erzeugung von Teilmodellen (Verzweigungsknoten) aus einem untersuchten aber nicht terminierten (Teil-)Modell

     Verzweigung muß geregelt werden:
     (a) Wie werden aus einem Modell seine Teilmodelle generiert?
     (b) Welcher Knoten wird als nächstes betrachtet?
        - Orientierung an den Schranken der aktiven (noch nicht überprüften) Knoten $\curvearrowright$ best-first-search
        - extern festgelegte Regeln

- schrankenunabhängige Regeln, wie z.B. LIFO $\curvearrowright$ depth-first-search
- dynamische Änderung der Auswahlregeln
  Idee: Zuerst LIFO, um schnell eine erste Schranke zu besitzen, dann schrankenabhängige Regeln

3. **Terminierung**
   - Nachweisen, daß der Lösungsbereich des aktuellen Teilmodells leer ist oder
   - eine Schranke für den Zielfunktionswert angeben, die schlechter ist als die aktuell beste Schranke

6 Gemischt-ganzzahlige Programmierung

6.1 Ansatz von Dakin

1. LP Relaxation lösen
2. Ist eine ganzzahlig geforderte Variable in der optimalen Lösung der Relaxation nicht ganzzahlig, dann ist ihre Ganzzahligkeit in den optimalen Lösungen der Teilmodelle durch geeignete ganzzahlige Schranken zu erreichen (Branch&Bound).

Algorithmus

1. Löse das relaxierte Problem mit dem Simplexalgorithmus
2. Ist eine ganzzahlig geforderte Variable x_i gebrochen, so verzweige in zwei Teilprobleme, denen jeweils eine zusätzliche Restriktion hinzugefügt wird. Splittet man den Wert von x_i in einen ganzzahligen und positiv gebrochenen Beitrag auf

   $$x_i = \lfloor x_i \rfloor + f_i$$

   so ergeben sich die zwei neuen Restriktionen zu

   $$x_i \leq \lfloor x_i \rfloor \; und \; x_i \geq \lfloor x_i \rfloor + 1$$

   bzw.

   $$x_i + x_{s_1} = \lfloor x_i \rfloor \; und \; x_i - x_{s_2} = \lfloor x_i \rfloor + 1$$

3. Führe für alle Verzweigungen folgendes durch:
   Reoptimiere das sich durch den Simplexalgorithmus ergebende Tableau mit der neuen Restriktion durch Aufnahme der neuen Schlupfvariable in die Basis und führe einen dualen Simplexschritt mit der neuen Restriktion als Pivotzeile durch

Eigenschaften der Verzweigung

- In jedem Knoten wird nach **einer** Verzweigungsvariablen in zwei Teilmodelle verzweigt
- Die Lösungsräume der Teilmodelle sind LP-Relaxationen mit einer zusätzlich eingeführten ganzzahligen Bedingung
- Die optimalen Zielfunktionswerte der relaxierten Lösungen sind obere Schranken für die ganzzahligen Teilmodelle
- Falls mehrere Variablen in einem Teilmodell die Ganzzahligkeit nicht erfüllen, so kann der Konflikt nach folgenden Kriterien beseitigt werden:
  - der zu erwartenden Zielfunktionswertverschlechterung
  - extern vorgegebene Prioriäten
  - Abweichung einer Variablen vom nächsten ganzzahligen Wert

7 Lagerhaltungsmodelle

verbale Formulierung

Ein Gut werde während eines endlichen Planungszeitraumes gelagert, der aus n Perioden bestehe. In jeder Periode werde das Lager einmal beliefert, und zwar jeweils zu Beginn der Periode. Außerdem trete in jeder Periode eine Nachfrage auf, die unmittelbar nach der Belieferung des Lagers in der betreffenden Periode befriedigt werde. Gesucht ist eine optimale Bestllpolitik, die die optimalen Bestellzeitpunkte und Bestellmengen festlegt.

- Aufbau: Beschaffung, Lagerung, Nachfragebefriedigung
- Beschränkungen:
  - **ein** zu lagerndes Gut
  - Lagergut diskret oder kontinuierlich
  - Lieferzeit = Zeit zwischen Bestellung und Lieferung

7.1 Klassisches Losgrößenmodell

Voraussetzungen

1. Lagerung **eines** Gutes während **unbegrenztem** Planungszeitraum
2. Nachfrage pro Zeiteinheit μ **konstant** (Abgangsrate)
3. **keine** Fehlmenge
4. Lieferzeit λ

Beschaffungskosten B

$$B = K \cdot \delta(Q) + c \cdot Q$$

$Q \geq 0$	Bestellmenge
$K \geq 0$	fixe Bestellkosten
$c \geq 0$	variable Beschaffungskosten pro Mengeneinheit (ME)
$\delta(Q)$	1 für $Q > 0$, 0 für $Q = 0$
$h \geq 0$	Lagerungskosten pro ME und ZE

Die Lagerhaltungskosten sind am kleinsten, falls die Lieferung genau dann eintrifft, wenn der Bestand Null wird. Da die Abgangsrate μ und die Lieferzeit λ konstant sind, ist es optimal, jeweils die gleiche Menge Q zu bestellen.

Die Zeit τ zwischen zwei Bestellungen ergibt sich zu

$$\tau = \frac{Q}{\mu}$$

C seien die Kosten pro Bestellperiode dividiert durch die Länge der Bestellperiode (BP). C gilt es zu minimieren.
Kosten/BP = Beschaffungskosten/BP + Lagerungskosten/BP = $K + c \cdot Q + h \cdot \tau \cdot \frac{Q}{2}$
$\frac{Q}{2}$ ist der mittlere Lagerbestand.

$$C = \frac{1}{\tau} \cdot (K + c \cdot Q + h \cdot \tau \cdot \frac{Q}{2})$$

bzw.

$$C = f(Q) = \underbrace{\frac{h}{2} \cdot Q}_{Lagerkosten} + \underbrace{K \cdot \frac{\mu}{Q}}_{Fixkosten} + \underbrace{c \cdot \mu}_{variable\ Kosten}$$

Um $C = f(Q)$ zu minimieren ist die Ableitung $f'(Q)$ gleich Null zu setzen:

Klassische Losgrößenformel

$$f'(Q) = \frac{h}{2} - \frac{K \cdot \mu}{Q^2} = 0 \ \curvearrowright \ Q^* = \sqrt{\frac{2 \cdot K \cdot \mu}{h}} \ und \ \tau^* = \sqrt{\frac{2 \cdot K}{h \cdot \mu}}$$

Minimale Gesamtkosten pro Zeiteinheit

$$C^* = \frac{h}{2} \cdot Q^* + K \cdot \frac{\mu}{Q^*} + c \cdot \mu = \sqrt{2Kh\mu} + c \cdot \mu$$

7.2 Klassisches Losgrößenmodell mit Fehlmengen

- Fehlmengenkosten (Strafkosten) **p**
- Fehlmengen werden als negativer Bestand interpretiert ↷
  Bei Eintreffen der Lieferung ist der Bestand **-v** ↷
  Bestand nach Wiederauffüllung ist **y = Q-v**
- zeitlicher Verlauf:
  - nach $\tau' = y/\mu$ ist der Bestand 0
  - nach $\tau'' = v/\mu$ ist der Bestand -v
  - Bestellperiode $\tau = \tau' + \tau''$
- Kosten der einzelnen Intervalle:
  - Während Intervall 1 (τ') ist der mittlere Bestand y/2 ↷
    Lagerungskosten: $h \cdot \tau' \cdot y/2$
  - Während Intervall 2 (τ'') ist der mittlere Fehlbestand v/2 ↷
    Fehlmengenkosten pro BP: $p \cdot \tau'' \cdot v/2 = p \cdot \tau'' \cdot (Q-y)/2$

Gesamtkosten pro Zeiteinheit

$$C = \frac{1}{\tau} \cdot \left(K + c \cdot Q + h \cdot \tau' \cdot \frac{y}{2} + p \cdot \tau'' \cdot \frac{Q-y}{2} \right)$$

Mit $\tau = Q/\mu$ und $\tau'/\tau = y/Q$ und $\tau''/\tau = (Q-y)/Q$ ergibt sich

$$C = f(Q,y) = \frac{h}{2} \cdot \frac{y^2}{Q} + \frac{p \cdot (Q-y)^2}{2 \cdot Q} + K \cdot \frac{\mu}{Q} + c \cdot \mu$$

Und daraus folgt

$$y^* = \sqrt{\frac{2K\mu}{h}} \cdot \sqrt{\frac{p}{h+p}}$$

und

$$Q^* = \sqrt{\frac{2K\mu}{h}} \cdot \sqrt{\frac{h+p}{p}}$$

und

$$\tau^* = \sqrt{\frac{2K}{h\mu}} \cdot \sqrt{\frac{h+p}{p}}$$

und schließlich

$$C^* = \sqrt{2Kh\mu} \cdot \sqrt{\frac{p}{h+p}} + c \cdot \mu$$

Da für $p \to \infty$ der Term $\frac{h+p}{p}$ gegen 1 konvergiert, ergeben sich dann die Lösungen für das Modell ohne Fehlmengen.

7.3 Dynamische Optimierung in der Lagerhaltung

- Planungszeitraum **endlich** $[t_a, t_e]$ in N gleichlange Perioden eingeteilt $\curvearrowright$ $\tau = t_j - t_{j-1}$
- Nachfragerate **nicht** konstant

Lagerbilanzgleichung

$$x_j = x_{j-1} + u_j - z_j$$

wobei

x_j	Lagerbestand am Ende von Periode j
u_j	zu Beginn von Periode j gelieferte Menge
z_j	Nachfrage in Periode j

Anfangs- und Endbestand x_0 und x_N sind als 0 vorgegeben.
Zur Optimierung der Bestellpolitik genügt es

$$\sum_{j=1}^{N}(K \cdot \delta(u_j) + c \cdot u_j + h \cdot \tau \cdot x_j)$$

zu minimieren. Die variablen Bestellkosten $c \cdot \sum u_j = c \cdot \sum z_j$ sind konstant, können also vernachläßigt werden.

Dynamisches Optimierungsmodell

Minimiere	$\sum_{j=1}^{N}(K \cdot \delta(u_j) + h \cdot \tau \cdot x_j)$	Kosten der Bestellpolitik
unter	$x_j = x_{j-1} + u_j - z_j,\ j = 1, \ldots, N$	Lagerbilanzgleichung
	$x_0 = x_N = 0$	Anfangs- und Endlagerbestand
	$x_j \geq 0,\ j = 1, \ldots, N-1$	keine Fehlmengen
	$u_j \geq 0,\ j = 1, \ldots, N$	positive Bestellmenge

3-stufiges Lagerhaltungsmodell

Variablen

x_i	Lagerbestand am Ende der Stufe i
y_i	Bestellmenge auf Stufe i
d_i	Nachfrage auf Stufe i

Lagerbilanzgleichungen

$$\begin{aligned} x_1 &= x_0 + y_1 - d_1 \\ x_2 &= x_1 + y_2 - d_2 \\ x_3 &= x_2 + y_3 - d_3 \end{aligned}$$

Kosten

Stufenkosten	$= c_i(x_{i-1}, y_i)$	abh. von Anfangsbestand u. Bestellmenge
Gesamtkosten	$= c_1(x_0, y_1) + c_2(x_1, y_2) + c_3(x_2, y_3)$	Summe der Stufenkosten

Berechnung der optimalen Bestellpolitik

1. Damit der Endzustand $x_3 = 0$ erreicht wird, ergibt sich

$$y_3^* = y_3(x_2) = d_3 - x_2$$

   Die Bestellmenge in Zeitintervall 3 ist also abhängig von der Menge, die nach Zeitintervall 2 übrig geblieben ist.

2. Nun ist abhängig von x_1 eine optimale Bestellung $y_2^* = y_2^*(x_1)$ zu bestimmen:

$$f_2^*(x_1) = \min_{y_2}[\ c_2(x_1, y_2) + f_3^*(x_1 + y_2 - d_2)]$$

   mit

$$f_3^*(x_2) = \min_{y_3 : x_2 + y_3 - d_3 = 0}[\ c_3(x_2, y_3)\ ] = c_3(x_2, d_3 - x_2)$$

   Es ist also das Minimum aus der Summe der Stufenkosten und der minimalen Kosten für den Restprozess über allen Bestellmöglichkeiten zu finden, abhängig von der Menge, die nach dem vorigen Zeitintervall übrig geblieben ist.

3.

$$f_1^*(x_0) = f_1^*(0) = \min_{y_1 \geq 0}[\ c_1(0, y_1) + f_2^*(0 + y_1 - d_1)\ ]$$

7.4 Verfahren von Wagner und Whitin

- Umkehrung des dynamischen Optimierungsmodells (siehe Kapitel 7.3 und 8)
- Lagerbilanzgleichung ist eindeutig nach x_{i-1} auflösbar $\curvearrowright$ Optimierungsrichtung umkehrbar

$$x_{i-1} = x_i - u_i + z_i$$

Bellmansche Funktionalgleichung

$$C_i^*(x_i) = \min_{0 \leq u_i \leq x_i + z_i}\{K \cdot \delta(u_i) + h \cdot \tau \cdot x_i + C_{i-1}^*(x_{i-1})\}$$

$$C_0^*(x_0) = 0$$

$C_n^*(x_n = 0)$ ist das Minimum der relevanten Kosten.

Diese Rekursion kann numerisch gelöst werden. Das Verfahren von Wagner und Whitin nutzt aber die **spezielle Struktur** der Rekursion zur Aufwandsminderung:

- Für ein optimales Lösungspaar (x^*_{i-1}, u^*_i) gilt entweder

$$x^*_{i-1} = 0 \ \wedge \ u^*_i > 0 \ \textit{oder} \ x^*_{i-1} > 0 \ \wedge \ u^*_i = 0$$

  Das heißt also, daß nur dann etwas bestellt wird, wenn das Lager leer ist.

- Eine optimale Bestellung u^*_i in der i-ten Periode kann nur einen Wert annehmen, der sich aus der Summe der Abgänge ergibt, also

$$u^*_i \in \{0 \ , \ z_i \ , \ z_i + z_{i+1} \ , \ \ldots \ , z_i + \ldots + z_n\}$$

Damit läßt sich die Bellmansche Funktionalgleichung vereinfachen:

$$\begin{aligned} C^*_0 &= \ 0 \\ C^*_1 &= \ p_1(1) = K \\ C^*_i &= \ p_i(\kappa^*_i) = \min_{k=1,\ldots,i} \{p_i(k)\} \end{aligned}$$

mit

$$p_i(k) = \ K + h \cdot \tau \cdot \sum_{i=k}^{i} \sum_{l=i+1}^{i} l \cdot z_i + C^*_{k-1}$$

Wird das Minimum für mehr als einen Index angenommen, so sei κ^*_i irgendeiner dieser Indizes.

Algorithmus

1. Funktionalgleichung in **Vorwärtsrechnung** lösen $\curvearrowright$
   C^*_i und κ^*_i ermitteln

2. Optimale Bestellmengen durch **Rückwärtsrechnung** bestimmen.

$$s = \kappa^*_n \ \curvearrowright \ u^*_s = z_s + \ldots + z_n \ \wedge \ u_{s+1} = \ldots = u_n = 0$$

$$m = \kappa^*_{s+1} \ \curvearrowright \ u^*_m = z_m + \ldots + z_{s-1} \ \wedge \ u^*_{m+1} = \ldots = u^*_{s-1} = 0$$

Diese Rechnung kann nun noch einmal vereinfacht werden:

- Für κ^*_i gilt: $\kappa^*_1 \leq \ldots \leq \kappa^*_n \curvearrowright$
  Bei Lösung der Funktionalgleichung muß nur das Minimum der Werte $p_i(\kappa^*_{i-1}), \ldots, p_i(i)$ gebildet werden, es kann also auf $p_i(1), \ldots, p_i(\kappa^*_{i-1} - 1)$ verzichtet werden

- Die $p_i(k)$ genügen der folgenden Rekursion:

$$p_i(k) = p_{i-1}(k) + (i-k) \cdot h \cdot \tau \cdot z_i, \ k = 1, \ldots, i-1$$

$$p_i(i) = K + C^*_{i-1}$$

8 Dynamische Optimierung

verbale Formulierung

In der dynamischen Optimierung werden Probleme betrachtet, die in einzelne "Stufen" zerlegt werden können, so dass die Gesamtoptimierung durch eine "stufenweise Optimierung" ersetzt werden kann. Diese Vorgehensweise bietet sich vor allem dann an, wenn es um die optimale Steuerung wirtschaftlicher, technischer oder anderer in der Zeit ablaufender Prozesse geht, wo die Stufen den einzelnen Zeitperioden entsprechen.

Problemdefinition

Betrachtung eines Systems während eines endlichen, in n Perioden eingeteilten Planungszeitraum

x_i	Zustandsvariable (Zustand des Systems am Ende von Periode i)
y_i	Entscheidungsvariable, Steuervariable (Entscheidung in Periode i)
X_i	Zustandsbereich (Alle möglichen Zustände am Ende von Periode i)
$Y_i(x_{i-1})$	Steuerbereich (Alle möglichen Entscheidungen in Periode i)
$c_i(x_{i-1}, y_i)$	Kosten der Periode i
$t(x_{i-1}, y_i)$	Zustandsübergang von x_{i-1} in x_i
$(x_0, \ldots, x_n)$	Zustandsfolge
$(y_1, \ldots, y_n)$	Entscheidungspolitik, Steuerung

Optimierungsproblem

$$
\begin{aligned}
&\text{Minimiere} && \sum_{i=1}^{n} c_i(x_{i-1}, y_i) \\
&\text{unter} && x_i = t_i(x_{i-1}, y_i)\ ,\ i = 1, \ldots, n \\
& && x_0 = x_a \\
& && x_i \in X_i\ ,\ i = 1, \ldots, n \\
& && y_i \in Y_i(x_{i-1})\ ,\ i = 1, \ldots, n
\end{aligned}
$$

Problem $P_i(x_{i-1})$

Bei gegebenen Funktionen c_i und t_i und gegebenen Zustands- und Steuerbereichen hängt das Optimierungsproblem (und damit auch dessen Lösungen) nur vom Anfangszustand x_0 ab. Dieses Problem wird mit $P_1(x_0)$ bezeichnet.

$P_i(x_{i-1})$ ist das Problem, welches nur die Perioden $j, \ldots, n$ umfaßt und lediglich von seinem Anfangszustand x_{i-1} abhängt.

Für $P_i(x_{i-1})$ ist $(y_i^*, \ldots, y_n^*)$ optimale Politik mit minimalen Kosten $f_i^*(x_{i-1}) \curvearrowright$
Für $P_{i+1}(x_i^*)$ ist $(y_{i+1}^*, \ldots, y_n^*)$ optimale Politik mit $x_i^* = t(x_{i-1}, y_i^*)$ und Kosten $f_{i+1}(x_i)$

Bellmansches Optimalitätsprinzip

Seien $(y_1^*, \ldots, y_n^*)$ eine optimale Politik für $P_1(x_0)$ und x_{i-1}^* der Zustand zu Beginn von Periode i. Dann ist $(y_i^*, \ldots, y_n^*)$ eine optimale Politik für $P_i(x_{i-1})$.

Die Entscheidungen in den Perioden i bis n des n-periodigen Problems $P_1(x_0)$ sind bei gegebenen Zustand x_{i-1} unabhängig von den Entscheidungen in den Perioden 1 bis i-1.

Bellmansche Funktionalgleichung

$$f_i^*(x_{i-1}) = \min_{y_i \in Y_i(x_{i-1})} \{c_i(x_{i-1}, y_i) + f_{i+1}^*(t_i(x_{i-1}, y_i))\} \ , \ x_{i-1} \in X_{i-1}$$

$$f_{n+1}^*(x_n) = 0 \ , \ x_n \in X_n$$

Die Bellmansche Funktionalgleichung stellt eine Beziehung zwischen zwei aufeinander folgenden Wertfunktionen f_i^* und f_{i+1}^* dar und erlaubt es, bei bekanntem f_{i+1}^* die Funktion f_i^* und damit jeweils für eine weitere Periode die Wertfunktion zu bestimmen.

Umkehrung des Rechenverlaufs

Erst Vorwärts-, dann Rückwärtsrechnung

Der Anfangszustand x_{i-1} einer Periode ist eindeutig bestimmt durch den gegebenen Endzustand x_i und die gegebene Entscheidung y_i in gleicher Periode, also

$$x_{i-1} = t_i(x_i, y_i)$$

Entsprechend gilt für die Summanden der Zielfunktion $c_i(x_i, y_i)$. Insgesamt ergibt sich bei Umkehrung das neue **dynamische Optimierungsproblem**:

$$\begin{array}{ll} \text{Minimiere} & \sum_{i=1}^{n} c_i(x_i, y_i) \\ \text{unter} & x_{i-1} = t_i(x_i, y_i) \ , \ i = 1, \ldots, n \\ & x_{n+1} = x_e \\ & x_i \in X_i \ , \ i = 1, \ldots, n \\ & y_i \in Y_i(x_i) \ , \ i = 1, \ldots, n \end{array}$$

Monotone Separabilität

Die Funktion z(x,y) heißt **separabel**, falls gilt:
Es existieren Verknüpfungsoperatoren $\circ$ und Zerlegungsfunktionen g_i, so daß z(x,y) darstellbar ist als

$$z(x, y) = g_1(c_1(x_0, y_1), \ldots, c_n(x_{n-1}, y_n))$$

mit

$g_n(c_n(x_{n-1}, y_n) = c_n(x_{n-1}, y_n)$

$g_i(c_i(x_{i-1}, y_i), \ldots, c_n(x_{n-1}, y_n)) = c_i(x_{i-1}, y_i) \circ g_{i+1}(c_{i+1}(x_i, y_{i+1}), \ldots, c_n(x_{n-1}, y_i))$

$$\underbrace{c_i \circ g_{i+1}(c_{i+1}, \ldots, c_n)}_{g_i(c_i, \ldots, c_n)}$$

Sind die Funktionen $g_i = c_i \circ g_{i+1}$ für jeden zulässigen Wert von c_i **monoton wachsend** in g_{i+1}, so gilt

$$\min_{y_i,\ldots,y_n} g_i(c_i\ ,\ g_{i+1}) = \min_{y_i} g_i(c_i\ ,\ \min_{y_{i+1},\ldots,y_n} g_{i+1}) = \min_{y_i} (c_i \circ \min_{y_{i+1},\ldots,y_n} g_{i+1})$$

Gilt die **monontone Separablilität**, so gilt auch die Bellmansche Funktionalgleichung des Dynamischen Optimierungsproblems.

$$f^*_{n+1}(x_n) = 0$$

$$f^*_i(x_{i-1}) = \min_{y_i} c_i(x_{i-1}, y_i) \circ f^*_{i+1}(t_i(x_{i-1}, y_i))$$

Bemerkung

Dynamische Prgrammierung kann wie auch Branch&Bound im Gegensatz zu vielen anderen OR-Verfahren, die spezielle Problemstrukturen erfordern, auf viele unterschiedliche Fragestellungen angewandt werden. Dabei muss man die jeweiligen Gleichungen selbst herleiten.
Besonders beliebt ist die Anwendung des Dynamischen Programmierens bei diskreten Problemen. Dies ist weniger darauf zurückzuführen, dass es hier besonders effizient ist, sondern darauf, dass die zu Lösung ganzzahliger Probleme zur Verfügung stehenden Methoden besonders ineffizient sind.

9 Nichtlineare Optimierung

verbale Formulierung

Ein nichtlineares Optimierungsproblem liegt vor, wenn die Zielfunktion und die in den Nebenbedigungen vorkommenden Funktionen der Entscheidungensvariablen nicht mehr sämtlich lineare Funktionen darstellen. Nichtlineare Optimierungsprobleme treten beispielsweise in der Produktionsplanung oder beim Transport von Gütern auf, wenn die Verkaufspreise oder Produktions- bzw. Transportkosten pro Mengeneinheit nicht mehr konstant sind. Dies führt auf Optimierungsprobleme mit nichtlinearer Zielfunktion. Die durch ökonomische Sachverhalte bedingte Restriktionen, wie z.B. Kapazitätsbeschränkungen für Produktionsfaktoren im Beschaffungs- oder Produktionsbereich, sind in der Regel linear. Nichtlinearitäten in den Nebenbedingungen, die erheblich schwieriger zu behandeln sind

als nichtlineare Zielfunktionen, treten etwa in Form von Materialbilanzgleichungen- und ungleichungen bei technischen Prozessen auf (z.B. in der chemischen Verfahrenstechnik), wenn die Endproduktmengen nichtlinear von den Ausgangsproduktmengen abhängen.
Eine besondere Schwierigkeit nichtlinearer Optimierungsprobleme besteht darin, dass man (bei Minimierungssaufgaben) zwischen lokalen bzw. relativen und globalen bzw. absoluten Minima unterscheiden muss. Die Modellierung und Formulierung nichtlinearer Zusammenhänge ist außerdem oft aufwendig, und einfachere lineare Modelle stellen häufig gute Annäherungen an die Realität dar. Ferner erfordern Verfahren zur Lösung nichtlinearer Optimierungsprobleme in der Regel einen großen Rechenaufwand, während für lineare Optimierungsaufgaben mit der Simplexmethode und deren Modifikation auch für sehr große Problme leistungsfähige Lösungsverfahren zur Verfügung stehen.

9.1 Konvexe Optimierung und Kuhn-Tucker-Theorie

Konvexe Funktion

Sei $M \subseteq \mathbb{R}^n$ konvex. Dann heißt eine Funktion $f : M \to \mathbb{R}$ konvex, wenn für je zwei Punkte $x_1, x_2 \in M$ und alle $\lambda \in (0, 1)$ gilt:

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)$$

Anschaulich: Alle Funktionswerte auf der Verbindungsgeraden zwischen $(x_1, f(x_1))$ und $(x_2, f(x_2))$ liegen oberhalb der Funktion.

Konvexität → lokales Optimum gleich globales Optimum

Ist $f : R^n \to R$ auf $M \subseteq R^n$ konvex, dann ist jedes lokale Minimum von f gleich dem globalen Minimum von f auf M.

Beweisidee: Indirekter Beweis
Annahme: Sei x^* lokales Optimum von f und x' so, dass $f(x') < f(x^*)$.
Da x^* lokales Optimum ist, existiert eine ϵ-Hülle um x^*, in der kein Funktionswert kleiner ist als der von x^*:

$$\exists U_\epsilon(x^*),\ \epsilon > 0,\ \textit{so dass}\ \forall x \in U_\epsilon(x^*) \cup M :\ f(x) \geq f(x^*)$$

Sei $x'' \in U_\epsilon(x^*)$ ein Punkt aus der ϵ-Hülle, der auf der Verbindungsgeraden von x^* und x' liegt. Da M konvex ist, liegen alle Verbindungspunkte in M.
Es existiert also ein $\lambda \in (0, \frac{\varepsilon}{|x^* - x'|})$, so dass

$$x'' = (1 - \lambda) \cdot x^* + \lambda \cdot x'$$

Also kann $f(x'')$ wiefolgt abgeschätzt werden:

$f(x'')$	$=$	$f((1 - \lambda)x^* + \lambda x')$	Punkt auf der Verbindungsgeraden
	$\leq$	$(1 - \lambda)f(x^*) + \lambda f(x')$	f ist konvex auf M
	$<$	$(1 - \lambda)f(x^*) + \lambda f(x^*)$	Annahme $f(x') < f(x^*)$
	$=$	$f(x^*)$	$(1 - \lambda) + \lambda = 1$, Widerspruch zum lok. Minimum

Standardproblem der Nichtlinearen Optimierung

Minimiere	$f(x),\ x \in R^n$
unter	$g_i(x) \leq 0\ \forall i = 1, \ldots, m$

$f(x)$ und/oder $g_i(x)$ sind nichtlinear.

Gradient

Seien f und g_i differenzierbar. Dann existiert der Gradient $\nabla f(x^*)$ von f im Punkt x^* und:

1. Der Gradientenvektor steht senkrecht auf der Tangentialhyperebene in x^*
2. Der Funktionswert f(x) wächst in Richtung des Gradienten
3. Falls $g_i(x^*) = 0$, so zeigt $\nabla g_i(x^*)$ aus Z hinaus.

Die lokale Optimalität eines Punktes x^* bedeutet also:

1. x^* ist zulässig
2. x^* muss in einer gewissen Umgebung bester Punkt sein $\curvearrowright$
   Ist x^* Randpunkt, so zeigt jede verbessernde Richtung aus Z hinaus

"verbessernde Richtung"

- Die Richtung des **steilsten Abstiegs** in x^* ist $-\nabla f(x^*)$.
  Jede verbessernde Richtung bildet mit $-\nabla f(x^*)$ einen Winkel $< 90°$.
- Sei I die Menge der Indizes der in x^* aktiven Restriktionen $\curvearrowright$ Der **Kegel**, der durch die Gradienten der aktiven Restriktionen aufgespannt wird, ergibt sich zu

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

- Liegt $-\nabla f(x^*)$ im Kegel $K(x^*)$, dann zeigt jede verbessernde Richtung aus Z hinaus.
- $-\nabla f(x^*)$ liegt genau dann in $K(x^*)$, wenn gilt:

$$\forall i \in I\ \exists u_i \geq 0 :\ -\nabla f(x^*) = \sum_{i \in I} u_i \cdot \nabla g_i(x^*)$$

Kuhn-Tucker-Bedingung (KTB)

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

$$u_i \geq 0,\ g_i(x^*) \leq 0$$

- **hinreichende KTB:**
  Seien f und g_i stetig differenzierbare und konvexe Funktionen, dann gilt:
  Existieren in einem Punkt x^* nichtnegative Skalare $u_i \geq 0$ für alle aktiven Restriktionen $i \in I$, so dass die Kuhn-Tucker-Bedingung erfüllt ist, dann ist x^* ein globales Minimum.

- Fordert man die Konvexität nicht, so liegt in x^* ein lokales Minimum vor.

- **notwendige KTB:**
  Seien f und g_i stetig differenzierbare Funktionen, nicht notwendig konvex, dann gilt:
  Ist x^* ein lokales Minimum, so existieren $u_i \geq 0\ \forall i \in I$, so dass die KTB erfüllt sind.

  Informell:
  Ist die Konvexität nicht verlangt, aber die lineare Unabhängigkeit der Gradienten der in x^* aktiven Restriktionen, so kann man zu jedem lokalen Minimum eine Lösung der KTB finden.

- **Spezialfall:**
  Ist x^* innerer Punkt von Z, so ist $I = \emptyset$ und die KTB ergeben sich zu:

$$\nabla f(x^*) = 0,\ g_i(x^*) < 0$$

Modifizierte Kuhn-Tucker-Bedingung

Da man bei der Suche nach Punkten, die die Kuhn-Tucker-Bedingung erfüllen, nicht weiß, welche Restriktionen aktiv sind, ist folgende modifizierte Form der KTB hilfreich und äquivalent:

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

$$u_i \cdot g_i(x^*) = 0$$

$$u_i \geq 0,\ g_i(x^*) \leq 0$$

Lagrange-Funktion

Die Funktion

$$L(x, u) = f(x) + u^T \cdot g(x)$$

$$wobei\ g(x)^T = (g_1(x), ..., g_m(x)),\ x \in \mathbb{R}^n,\ u \in \mathbb{R}^m$$

heißt Lagrange-Funktion zum Standardproblem der Nichtlinearen Optimierung.

Ein Vektor $(x_0, u_0)^T \in \mathbb{R}^{m+n}$ mit $x_0, u_0 \geq 0$ heißt **Sattelpunkt** von L(x,u), wenn für alle $x \geq 0$ und alle $u \geq 0$ gilt

$$L(x_0, u) \leq L(x_0, u_0) \leq L(x, u_0)$$

Notwendige Bedingungen für einen Sattelpunkt von L in $(x_0, u_0)^T$:

$$L_x(x_0, u_0) \geq 0$$

$$L_u(x_0, u_0) \leq 0$$

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

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

$$x_0, u_0 \geq 0, mit\ L_u(x_0, u_0) = g(x)$$

Zusammenhang zwischen NLP und Lagrange-Funktion:
Ist $(x_0, u_0)^T$ mit $x_0, u_0 \geq 0$ ein **Sattelpunkt von L**, so ist x_0 eine **optimale Lösung** des Standardproblems der Nichtlinearen Optimierung.

Beweisidee:

Wegen der ersten Ungleichung in der Sattelpunktdefinition gilt:

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

Dies ist nur möglich, wenn $g(x_0) \leq 0$ ist **(Zulässigkeit)**. Wählt man $u = 0$, so ergibt sich $u_0^T \cdot g(x_0) \geq 0$. Wegen $g(x_0) < 0$, ist aber auch $u_0^T \cdot g(x_0) \leq 0$. Es gilt also insgesamt:

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

Aufgrund der zweiten Ungleichung der Sattelpunktdefinition gilt:

$$f(x_0) + u_0^T \cdot g(x_0) \leq f(x) + u_0^T \cdot g(x)$$

Und mit $u_0^T \cdot g(x_0) = 0$ und $u_0^T \cdot g(x) < 0$ ergibt sich:

$$f(x_0) \leq f(x) + u_0^T \cdot g(x) \leq f(x)$$

x_0 ist also zulässig, d.h eine optimale Lösung.

Die Umkehrung des Satzes gilt nur, wenn die Konvexität und die sogenannte Slater-Bedingung erfüllt ist:
Der Punkt x erfüllt die **Slater-Bedingung**, wenn x innerer Punkt bezüglich aller nichtlinearen Restriktionen ist, das heißt also das $g_i(x) < 0$ gilt für alle nichtlinearen Restrikionen g_i.

Erfüllt das Standardmodell der Nichtlinearen Optimierung die Slater-Bedingung, so sind die KTB notwendiges und hinreichendes Kriterium für eine optimale Lösung des Modells.

Ist L(x,u) konvex in x und konkav in u, so sind die KTB notwendiges und hinreichendes Kriterium für einen Sattelpunkt von L.

9.2 Quadratisches Programmieren

Das Modell

Minimiere	$f(x) = c^T x + \frac{1}{2} x^T Q x$
so dass	$Ax \leq b$
	$x \geq 0$

Also: Zielfunktion quadratisch, Nebenbedingungen linear. Ist Q symmetrisch und positiv semidefinit, so ist f(x) konvex.

Die zugehörige **Lagrange-Funktion** ergibt sich zu

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

L(x,u) ist konvex in x und konkav in u $\curvearrowright$
KTB sind notwendiges und hinreichendes Kriterium für eine optimale Lösung des Modells, die Slaterbedingungen sind hier nicht relevant

Kuhn-Tucker-Bedingung

$$L_x(x,u) = c^T + Qx + A^T u \geq 0$$
$$L_u(x,u) = Ax - b \leq 0$$
$$x_0 \cdot L_x(x,u) = 0$$
$$u_0 \cdot L_u(x,u) = 0$$

9.2.1 Algorithmus von Wolfe

Voraussetzungen

1. Zielfunktion quadratisch, Nebenbedingungen linear, $x \geq 0$
2. Q symmetrisch, positiv semidefinit (Zielfunktion konvex)

```
1.  Formuliere KTB für das Modell
2.  Füge den linearen Bedingungen Schlupfvariablen hinzu
3a. Falls zulässige Basislösung vorliegt unter Beachtung der KTB,
    so ist die Lösung optimal
3b. Falls nicht, so füge wo notwendig Hilfsvariablen hinzu
    und iteriere mit der M-Methode
```

Beschränkung:
Ist x_j in der Basislösung, so darf s_{1j} nicht aufgenommen werden und umgekehrt.

9.3 Verfahren der Nichtlinearen Programmierung

Minimierung ohne Restriktionen

Prinzipielles Vorgehen:

- Problem: Minimiere f(x)
- Gegeben: $x^k \in R^n$ als zulässiger Iterationspunkt
- Gesucht: $x^{k+1} \in R^n$ mit $f(x^{k+1}) < f(x^k)$

Idee:

- Bestimme Richtung $d^k \in R^n$, in der f fällt
- Löse $min\{f(x^k + \mu \cdot d^k) \mid \mu > 0\}$ (Schrittweitenbestimmung)
- Setze $x^{k+1} = x^k + \mu \cdot d^k$ mit optimaler Schrittweite μ^k

Gradientenverfahren

- Verfahren des steilsten Abstiegs
- Richtung des steilsten Abstiegs ist $-\nabla f(x^k)$
- Gegeben sind also x^k und $d^k = -\nabla f(x^k)$
- Bestimme $\mu^k = arg\ min\{f(x^k + \mu d^k) \mid \mu \geq 0\}$ und $x^{k+1} = x^k + \mu^k d^k$
- Abbruchbedingung: $|\nabla f(x^k)| < \varepsilon$
- Es gilt immer $f(x^{k+1}) < f(x^k)$, aber das Konvergenzverhalten in der Nähe des Optimums ist oft schlecht

9.3.1 Mehrdimensionales Newtonverfahren

- f sei zweimal stetig differenzierbar
- Quadratische Näherung von $f(x)$ in x^k (mehrdimensionale **Taylorentwicklung**)

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

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

$$x^{k+1} = x^k - H^{-1}(x^k) \cdot \nabla f(x^k) \qquad \left(= x^k - \frac{f'(x^k)}{f''(x^k)} \right)$$

- Ein quadratisches Problem $f(x) = 1/2x^T Ax - b^T x + c$ ist lösbar in einem Iterationsschritt, da
$$x^* = x^0 - A^{-1}(Ax^0 - b) = A^{-1}b$$
- Falls x^0 hinreichend nahe bei x^* liegt, so konvergiert das Newton-Verfahren quadratisch gegen x^*, d.h. es gilt
$$||x^{k+1} - x^*|| < \gamma ||x^k - x^*||^2,\ 0 < \gamma < 1$$
- **Vorteile:**
  - Quadratische Konvergenz
- **Nachteile:**
  - Notwendigkeit der Berechnung der 2. Ableitung in jedem Schritt
  - Lösen des Gleichungssystems ist aufwendig
  - $H(x^k)$ kann singulär oder schlecht konditioniert sein
  - Das Verfahren ist nur lokal konvergent

9.3.2 Verfahren zulässiger Richtungen

verbale Formulierung

Das Grundprinzip des Verfahrens ist wie folgt: In jedem Optimierungsschritt werden zwei Teilprobleme gelöst. Zunächst wird eine zulässige Optimierungsrichtung bestimmt, welche zudem die Zielfunktion verbessert (verbessernde, zulässige Richtung). Das zweite Teilproblem besteht dann in der Festlegung der Schrittweite entlang dieser Richtung. Da die Richtung bereits vorgegeben ist, ist die Bestimmung der Schrittweite ein eindimensionales Line-Search-Problem, welches mit Standardmethoden gelöst werden kann.

Explizite Berücksichtigung der Nebenbedingungen im Verlaufe des Verfahrens.

Problem:

- Minimiere $f(x)$, und $x^k \in Z$, $Z = \{x \in R^n \mid g_i(x) \leq 0\}$

Vorgehen:

- Bestimme Vektor $d^k \in R^n$ mit
  1. $x^k + \mu d^k \in Z\ \forall \mu \in R$ mit $0 \leq \mu < \varepsilon$
  2. $f(x^k + \mu d^k) < f(x^k)\ \forall \mu$
  3. Schrittweitenbestimmung $\mu^k \in [0, \varepsilon]$
  4. $x^{k+1} = x^k + \mu^k d^k$

d^k heißt **verbessernde zulässige Richtung**

Verfahren von Zoutendijk

Sei $x \in Z$ und $I = \{i \mid g_i(x) = 0\}$. Wenn $d \in R^n$ die Ungleichungen

$$\nabla f(x)^T \cdot d < 0 \; und \; \nabla g_i(x)^T \cdot d < 0 \; \forall i \in I$$

erfüllt, so ist d eine verbessernde und zulässige Richtung.

Für ein gegebenes x läßt sich eine derartige Richtung d durch Lösen eines LP mit den Variablen d_j und z bestimmen:

Minimiere	z
unter	$\nabla f(x)^T \cdot d - z \leq 0$
	$\nabla g_i(x)^T \cdot d - z \leq 0$
	$-1 \leq d_j \leq 1$

Es gilt:
Falls $z < 0$, so ist d eine verbessernde zulässige Richtung.
Falls $z = 0$, so ist x ein Kuhn-Tucker-Punkt.

9.3.3 Minimierung mit Nebenbedingungen

Strafkosten- und Barriereverfahren:

Bei den Methoden der Straffunktionen und der Barrierefunktionen wird das zu lösende restringierte Optimierungsproblem durch unrestringierte Probleme approximiert, in dem man entweder zur Zielfunktion f einen "Strafterm" addiert, der das Verletzen der Restriktionen mit hohen Kosten bestraft, oder auf dem Rand von Z eine "Barriere" errichtet, die das Verlassen von Z verhindert.

- Nebenbedingungen derart in Zielfunktion einbeziehen, daß unzulässige Punkte Strafkosten verursachen
- Lösen des unrestringierten Problems mit vorangegangenen Methoden
- Strafkostenverfahren (Penaltyverfahren):
  $\{x^k\}_{k=0,1,...}$ Folge unzulässiger Punkte (Näherung an den Lösungsraum von aussen)
- Barriereverfahren:
  $\{x^k\}_{k=0,1,...}$ Folge zulässiger Punkte

Strafkostenfunktion:

$$S(x) = \sum_{i=1}^{m} (max\{0, g_i(x)\})^p$$

Es gilt:

$$S(x) \begin{cases} = 0 & \text{, für } x \in Z \\ > 0 & \text{, für } x \notin Z \end{cases}$$

Häufig wird $p = 2$ verwendet, da durch die Quadrierung eine Glättung der Straffunktion erreicht wird. Je stärker ein Punkt Restriktionen verletzt, desto größer ist der Wert der Strafkostenfunktion. Punkte innerhalb des zulässigen Bereichs erzeugen keine Strafkosten.

Folge von Strafkostenfunktionen:

Eine Folge von Funktionen $S_r(x),\ r \geq 0$ heißt Folge von Strafkostenfunktionen, falls gilt:

- $\mu_r \cdot S_r(x)$ ist stetig
- $\mu_r \cdot S_r(x) = 0$ für alle $x \in Z$
- $\mu_{r+1} \cdot S_{r+1}(x) > \mu_r \cdot S_r(x)\ \forall r \geq 0$ und $\forall x \notin Z$ (monoton wachsend)
- $\mu_r \cdot S_r(x) \to \infty$ für $r \to \infty$ und $x \notin Z$

Lösungsansatz:

- Minimiere $\{f(x) + \mu_r \cdot S_r(x)\}_{r=0,1,\ldots}$
- Parameter μ klein $\curvearrowright$ geringes Gewicht der Restriktionen
- Parameter μ groß $\curvearrowright$ Verletzung der Restriktionen wird hart bestraft
- Vorgehen: Mit kleinem μ beginnen und dann μ vergrößern, um sich Z zu nähern

Algorithmus:

- Wähle $\varepsilon > 0$, $\mu_0 \geq 0$, $\beta > 1$, $x_0 \in R^n$
- Setze r=0
- DoWhile($\mu_r S(x^r) > \varepsilon$)
  $x^{r+1} = arg\ min\{f(x) + \mu_r S(x)\}$
  $\mu_{r+1} = \beta\mu_r$
  $r = r + 1$
  EndDo