



Operations Research

Jürgen Pannek

Wintersemester 2015/16

Jürgen Pannek

Fachgebiet Dynamics in Logistics
Fachbereich 4 – Produktionstechnik

BIBA – Bremer Institut für Produktion und Logistik
Universität Bremen, Hochschulring 20, 28359 Bremen

Sprechstunde	Freitag 14 – 16 Uhr
Büro	Gebäude BIBA – Raum 1090
Telefon	+49 (0) 421 218–50190
E-Mail	panbiba.uni-bremen.de
www	http://www.dil.biba.uni-bremen.de

Ziel

- ▶ Untersuchung der Dynamik in logistischen Systemen
- ▶ Nutzung von Synergien zwischen vier Disziplinen

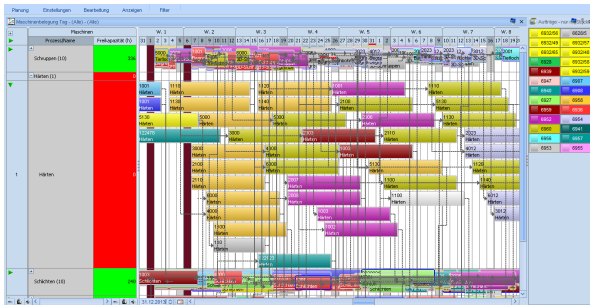
Strategie

- ▶ Nachhaltigkeit der Grundlagenforschung
- ▶ Wissenstransfer in Industrie
- ▶ Ausbildung auf Promotionsniveau



Vorlesung	Freitag 12-14 Uhr, HS 1010
Tutorium	Mittwoch 12-14 Uhr, GW1-HS H0070
Zielgruppe	Betriebswirtschaftslehre Wirtschaftswissenschaft Wirtschaftsingenieurwesen Produktionstechnik
Inhalt	Modelle, Methoden und Algorithmen mit besonderer Bedeutung in Theorie und Praxis
www	http://www.dil.biba.uni-bremen.de/education.html

- ▶ Warum ist die Reihenfolge bei der Produktionsplanung so schwierig?



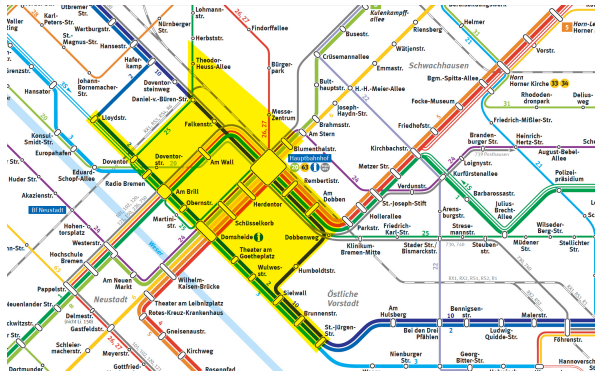
Quelle: www.planungstafel.net

- ▶ Wie belädt man ein Containerschiff, um den Aufwand fürs Entladen gering zu halten?



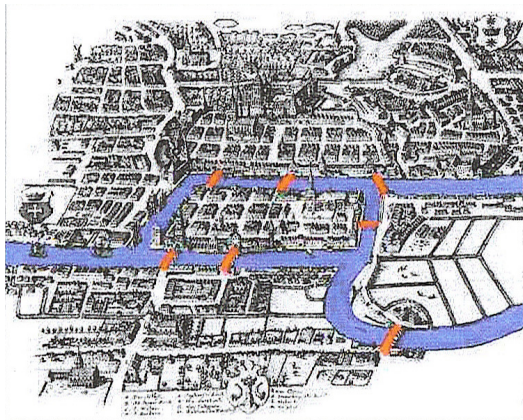
Quelle: www.wikipedia.org

- Wie erstellt man einen sinnvollen Fahrplan (Bahn, Bus, S-Bahn, ...)?



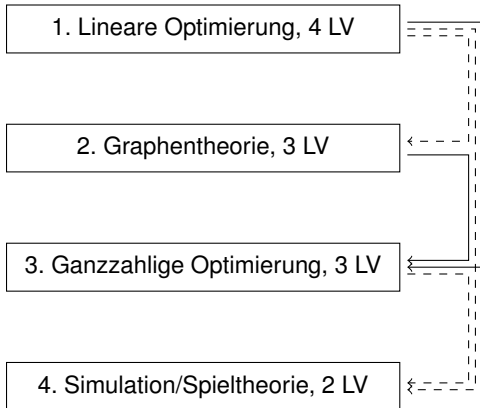
Quelle: www.vbn.de

- ▶ Welchen Nutzen hat der Leonhard Euler (1707–1783) für Google Street View?



Quelle: www.friderizianer.de

- ▶ Wie ernähre ich mich richtig und zugleich kosten-minimal bei Burger King?
- ▶ Wie simuliert man Warteschlangen in der Mensa und lässt sich dadurch die Wartezeit verkürzen?
- ▶ Drei Personen teilen sich ein Taxi, wie verteilt man die Fahrtkosten *fair*?
- ▶ Wie berechnet man die kürzeste Route, um alle Sehenswürdigkeiten Bremens zu besuchen?



- ▶ Optimierung wirtschaftswissenschaftlicher Prozesse (Pannek)
- ▶ Modeling, Analysis, and Control of Dynamics in Logistics (Becker, Freitag, Pannek)
- ▶ Komplexe Netzwerke in Produktion und Logistik (Becker)
- ▶ Computational Operations Management (Buer)
- ▶ Optimierung in Produktion und Logistik (Buer)
- ▶ Projekt Logistik I/II (Buer, Kopfer)
- ▶ Ökologische Transportplanung (Kopfer)
- ▶ Modellierung von Logistiksystemen (Kopfer)
- ▶ Transportplanung und Steuerung (Kopfer)
- ▶ ...

Leitung	Tobias Sprodowski
Termin	Mi 12-14 Uhr, GW1-HS H0070
Unterlagen	StudIP



Tobias Sprodowski

Bitte für Tutorien eines der folgenden Programme installieren:

- ▶ **Microsoft Excel** (27,-) mit Solver Add-In *oder*
- ▶ **Open Office Calc / Libre Office Calc** (kostenlos) *oder*
- ▶ **Google Drive – Tabellenkalkulation** (kostenlos)

Vorinstallieren

Software zum ersten Tutorium notwendig

Operations Research

J. Pannek

Organisatorisches

Termine

Inhalte

Literatur

Leistungsnachweis

11

- ▶ Werners (2013) **Grundlagen des Operations Research**, 3. Auflage, Springer.
- ▶ Domschke, Drexl (2005) **Einführung in Operations Research**, 6. Auflage, Springer.
- ▶ Gerdtz (2001) **Mathematische Optimierungsverfahren des Operations Research**, de Gruyter.
- ▶ Nickel, Stein, Waldman (2014) **Operations Research**, 2. Auflage, Springer.
- ▶ Suhl, Mellouli (2013) **Optimierungssysteme**, 3. Auflage, Springer.
- ▶ Neumann, Morlock (2005) **Operations Research**, 2. Auflage, Hanser Fachbuch.

Klausur Dauer 90 Minuten, maximal 90 Punkte

Termin 22.02.2016

Bestehen Mindestens 45 Punkte

Note 1,0 Mindestens 85 Punkte

Hilfsmittel Alles auSSer Telefon- und Internetjoker

Vorleistungen Keine

Vorbereitung

Bulimie Lernen sinnlos, besser kontinuierlich mitarbeiten

Operations Research ...

- ▶ ist praxisrelevant
- ▶ hat viele Anwendungsfelder
- ▶ bietet einen Methodenbaukasten
- ▶ vertieft eigene Rechenkenntnisse
- ▶ trainiert Umgang mit Schnittstellenproblemen

Teil I

Quantitative Entscheidungsunterstützung

Entscheidungsunterstützung

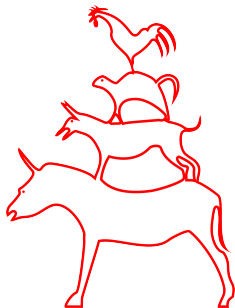
Beispiel Produktionsprogrammplanung

Mathematische Optimierung

Mathematisches Modell

Mathematisches Optimierungsproblem

Lineares Optimierungsproblem



Schritte zur Erfüllung von (Management-)Aufgaben:

1. **Planung / Entscheidungsvorbereitung**
2. Entscheidung
3. Durchführung
4. Kontrolle

Planen/Entscheiden:

- ▶ Ziel(e) identifizieren
- ▶ Alle Handlungsalternativen kennen
- ▶ Auswahl einer Handlungsalternativen, die das Ziel am besten erfüllt

Ziele des Operations Research (OR)

1. Vorbereitung möglichst **optimaler** Entscheidungen durch Anwendung mathematischer Methoden
2. Überführung eines realen Entscheidungsproblems durch ein **Optimierungs-** oder **Simulationsmodell**
3. Anwendung bzw. Entwicklung eines **Algorithmus** zur Lösung des Modells

Modell

Zweckorientiertes, vereinfachtes Abbild der Realität welches hinsichtlich der interessierenden Eigenschaften strukturähnlich ist.

Algorithmus

Handlungsvorschrift zur Lösung eines Problems

17 Entscheidungsunterstützung

Beispiel Produktionsprogrammplanung

Mathematische Optimierung

Zusammenfassung

Entscheidungsunterstützung

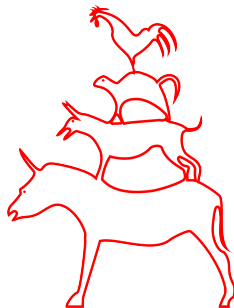
Beispiel Produktionsprogrammplanung

Mathematische Optimierung

Mathematisches Modell

Mathematisches Optimierungsproblem

Lineares Optimierungsproblem



Parameter:

Möbeltyp	Produktionsdauer (in Stunden)			Gewinn [GE]
	Maschine 1	Maschine 2	Arbeitszeit	
Stuhl	3,5	4	11	35
Bank	6	5	15	45
Tisch	8	6	20	65

Restriktionen:

- ▶ Maschine 1 ist für 120 Stunden pro Woche verfügbar.
- ▶ Maschine 2 ist für 100 Stunden pro Woche verfügbar.
- ▶ Es gibt 7 Tischler, die je 40 Stunden pro Woche arbeiten.
- ▶ Tische sollen höchstens die Hälfte aller produzierten Möbel ausmachen.

Frage

Wie viele Möbelstücke sollen wir (innerhalb einer Woche) produzieren, um unseren Wochengewinn zu maximieren?

Entscheidungen:

- ▶ Wie viele Stühle sollen produziert werden? (x_S)
- ▶ Wie viele Bänke sollen produziert werden? (x_B)
- ▶ Wie viele Tische sollen produziert werden? (x_T)

Gewinn:

$$\underbrace{z}_{\text{Zielfunktionswert}} = f(x) = 35x_S + \overbrace{45}^{\text{Parameter}} x_B + 65 \underbrace{x_T}_{\text{Entscheidungsvariable}}$$

Ziel: Sinnvolle Werte für x_S, x_B, x_T bei Gewinnmaximierung

Optimierungsvorschrift

$$\max z = 35x_S + 45x_B + 65x_T$$

u.d.N. $3.5x_S + 6x_B + 8x_T \leq 120$ (Maschine 1)

Nebenbedingungen $4x_S + 5x_B + 6x_T \leq 100$ (Maschine 2)

$$11x_S + 15x_B + 20x_T \leq 7 \cdot 40 \text{ (Arbeitszeit)}$$

$$x_T \leq x_S + x_B$$

$$x_S \geq 0$$

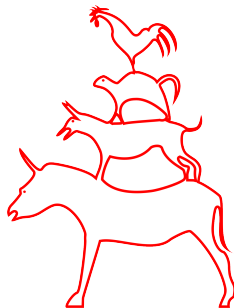
$$x_B \geq 0$$

$$x_T \geq 0$$

Entscheidungsunterstützung

Beispiel Produktionsprogrammplanung

Mathematische Optimierung
Mathematisches Modell
Mathematisches Optimierungsproblem
Lineares Optimierungsproblem



Operations Research

J. Pannek

Entscheidungsunterstützung

Beispiel Produktionsprogrammplanung

20 **Mathematische Optimierung**

Mathematisches Modell

Mathematisches Optimierungsproblem

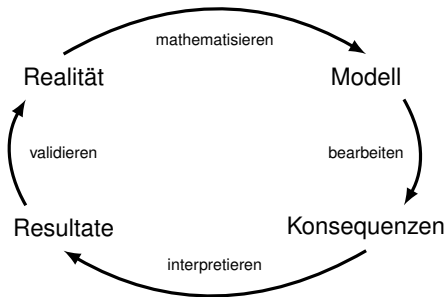
Lineares Optimierungsproblem

Zusammenfassung

Mathematisches Modell

Sammlung von Variablen und Beziehungen zwischen Variablen zur Beschreibung wichtiger Eigenschaften eines gegebenen Problems.

Wie konstruiert man ein mathematisches Modell?



Optimierung

Ermittlung derjenigen zulässigen Handlungsalternative, die einem vorgegebenem Ziel **am besten** von allen Alternativen entspricht.

Mathematisches Optimierungsproblem

$$\max\{f(\mathbf{x}) : \mathbf{x} \in S\}$$

1. \mathbf{x} heiSSt Vektor von **Entscheidungsvariablen**
↪ repräsentiert ein oder mehrere Alternativen
2. S enthält alle erlaubten \mathbf{x}
↪ wird durch **Restriktionen** eingeschränkt
3. $f(\mathbf{x})$ heiSSt **Zielfunktion**
↪ erlaubt Suche nach optimalem \mathbf{x}

Lösung

Eine Lösung ist eine Zuweisung von Werten zu allen Entscheidungsvariablen.

Möbel-Beispiel: $\bar{\mathbf{x}} = (8, 0, 5)$ oder $\hat{\mathbf{x}} = (3, 6, 27)$

Zulässige Lösung

Eine Lösung heißt zulässig, falls sie **alle** Restriktionen erfüllt.

Möbel-Beispiel: $\bar{\mathbf{x}} = (8, 0, 5)$ ist zulässig, dagegen ist $\hat{\mathbf{x}} = (3, 6, 27)$ unzulässig.

Wert/Zielfunktionswert

Der Wert einer zulässigen Lösung \mathbf{x} ist ihr Zielfunktionswert $z = f(\mathbf{x})$.

Möbel-Beispiel: Für $\bar{\mathbf{x}} = (8, 0, 5)$ ist der Zielfunktionswert

$$z = f(\bar{\mathbf{x}}) = f(8, 0, 5) = 35 \cdot 8 + 45 \cdot 0 + 65 \cdot 5 = 605$$

Optimale Lösung

Eine zulässige Lösung eines Maximierungsproblems heißt **optimal**, wenn ihr Wert **mindestens so groß** ist wie der jeder anderen zulässigen Lösung.

Möbel-Beispiel: Die Lösung $\mathbf{x}^* = (9.03225, 0, 9.03225)$ ist optimal, ihr Zielfunktionswert lautet 903.225.

Operations Research

J. Pannek

Entscheidungsunterstützung

Beispiel Produktionsprogrammplanung

Mathematische Optimierung

Mathematisches Modell

Mathematisches Optimierungsproblem

Lineares

Optimierungsproblem

Zusammenfassung

24

Lineares Optimierungsproblem

Ein mathematisches Optimierungsproblem

$$\max\{f(\mathbf{x}) : \mathbf{x} \in S\}$$

heißt **linear**, wenn

- ▶ die Entscheidungsvariablen **beliebig teilbar** ($\mathbf{x} \in \mathbb{R}^n$),
- ▶ die Parameter **deterministisch**, und
- ▶ Zielfunktion $f(\mathbf{x})$ und Nebenbedingungen **linear** sind.

Spezialfall: Lineare Gleichung

Eine Gleichung wird als **lineare Gleichung** bezeichnet, wenn gilt

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y) \quad \forall \alpha, \beta \in \mathbb{R}.$$

Warum Fokus auf LPs? es gibt effiziente Lösungsverfahren, hohe Verbreitung bei Unternehmen

Lineares Programm

Als **lineares Programm** in allgemeiner Form bezeichnen wir

$$\max f(\mathbf{x}) = c_1 x_1 + \dots + c_n x_n$$

$$\text{u.d.N.} \quad a_{i1} x_1 + \dots + a_{in} x_n \leq b_i, \quad \text{für } i = 1, \dots, m_1$$

$$a_{i1} x_1 + \dots + a_{in} x_n \geq b_i, \quad \text{für } i = m_1 + 1, \dots, m_2$$

$$a_{i1} x_1 + \dots + a_{in} x_n = b_i, \quad \text{für } i = m_2 + 1, \dots, m$$

$$x_j \geq 0, \quad \text{für einige oder alle } j = 1, \dots, n$$

Gegeben:

Zielfunktionskoeffizienten: c_1, c_2, \dots, c_n

Kapazitäten („rechte Seite“): b_1, b_2, \dots, b_m

Koeffizientenmatrix: $A^{m \times n} = (a_{ij}), i = 1, \dots, m; j = 1, \dots, n$

Gesucht:

Entscheidungsvariablen: x_1, x_2, \dots, x_n

Fragen zur Wiederholung:

- ▶ Was versteht man unter den Begriffen/Konzepten **Operations Research, Modell, Algorithmus, Entscheidungsunterstützung**?
- ▶ Welche Strukturen charakterisieren ein **Lineares Programm**?
- ▶ Wodurch zeichnet sich eine **optimale Lösung** aus?

Klausurrelevante Literatur

Domschke/Drexel: Kapitel 1.1, 1.2 und 2.1, oder Neumann/Morlock Kapitel 0 und 1.1

Ausblick

- ▶ Tutorium: Modellierung einfacher LP
- ▶ Wie lassen sich LP grafisch lösen?

Teil II

Lineare Optimierung I – Graphisches Lösungsverfahren und Normalform

Lineares Optimierungsproblem

Ein mathematisches Optimierungsproblem

$$\max\{f(\mathbf{x}) : \mathbf{x} \in S\}$$

heißt **linear**, wenn

- ▶ die Entscheidungsvariablen **beliebig teilbar** ($\mathbf{x} \in \mathbb{R}^n$),
- ▶ die Parameter **deterministisch**, und
- ▶ Zielfunktion $f(\mathbf{x})$ und Nebenbedingungen **linear** sind.

Spezialfall: Lineare Gleichung

Eine Gleichung wird als **lineare Gleichung** bezeichnet, wenn gilt

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y) \quad \forall \alpha, \beta \in \mathbb{R}.$$

↪ Warum Fokus auf LPs?

Lineares Programm

Als **lineares Programm** in allgemeiner Form bezeichnen wir

$$\max f(\mathbf{x}) = c_1 x_1 + \dots + c_n x_n$$

$$\text{u.d.N.} \quad a_{i1} x_1 + \dots + a_{in} x_n \leq b_i, \quad \text{für } i = 1, \dots, m_1$$

$$a_{i1} x_1 + \dots + a_{in} x_n \geq b_i, \quad \text{für } i = m_1 + 1, \dots, m_2$$

$$a_{i1} x_1 + \dots + a_{in} x_n = b_i, \quad \text{für } i = m_2 + 1, \dots, m$$

$$x_j \geq 0, \quad \text{für einige oder alle } j = 1, \dots, n$$

Gegeben:

Zielfunktionskoeffizienten: c_1, c_2, \dots, c_n

Kapazitäten („rechte Seite“): b_1, b_2, \dots, b_m

Koeffizientenmatrix: $A^{m \times n} = (a_{ij}), i = 1, \dots, m; j = 1, \dots, n$

Gesucht:

Entscheidungsvariablen: x_1, x_2, \dots, x_n

30 Wiederholung

Graphische Lösung
von LP

Eigenschaften
zulässiger Lösungen

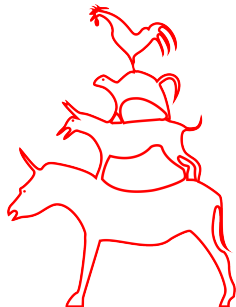
Normalform und Basis

Zusammenfassung

Graphische Lösung von LP
Beispiel Produktionsprogrammplanung
Algorithmus
Beispiel

Eigenschaften zulässiger Lösungen

Normalform und Basis
Standardform
Normalform
Basis
Zulässige Basislösung
Ecke



Problemstellung:

- ▶ Ein Unternehmen produziert **zwei Produkte** P_1 und P_2 .
- ▶ Deckungsbeitrag (pro Stück): P_1 liefert **3 GE**, P_2 liefert **4 GE**.

Restriktionen:

	Ressourcen pro Stück		verfügbare Ressourcen
	Produkt P_1	Produkt P_2	
Maschinenstunden (h)	3	2	1200
Rohstoff (ME)	5	10	3000
Arbeitskraft (h)	0	0.5	125

Frage

Welches Produktionsprogramm maximiert den Gesamtgewinn?

1. Modellierung der Problemstellung als LP:

$$\max z = f(\mathbf{x}) = 3x_1 + 4x_2$$

$$3x_1 + 2x_2 \leq 1200 \quad \text{Maschinen (h)} \quad \text{(I)}$$

$$5x_1 + 10x_2 \leq 3000 \quad \text{Rohstoffe (ME)} \quad \text{(II)}$$

$$0.5x_2 \leq 125 \quad \text{Arbeitszeit (h)} \quad \text{(III)}$$

$$x_1 \geq 0 \quad \text{(IV)}$$

$$x_2 \geq 0 \quad \text{(V)}$$

2. Welche Anzahl x_1 von P_1 und x_2 von P_2 soll produziert werden?

Algorithmus

Der Algorithmus (=Verarbeitungsvorschrift) sieht **fünf Schritte** vor:

1. Bestimme die Menge \mathbb{M} aller **zulässigen Lösungen**.
2. Zeichne **eine** Isogewinngerade.
3. Verschiebe diese Isogewinngerade solange **parallel in Richtung des Gradienten** von $f(\mathbf{x})$ **bis durch weiteren Verschieb kein weiterer zulässiger Punkt** erreicht werden kann.
4. Berechne die Koordinaten von $\mathbf{x}^* = (x_1^*, x_2^*)$ durch **Auflösen des Gleichungssystems der aktiven Restriktionen**.
5. Berechne $z^* = f(x_1^*, x_2^*)$, z^* ist **der optimale Zielfunktionswert**.

Graphische Darstellung von Restriktionen

Führe die folgenden Schritte für jede Restriktion des LP durch:

1. Fasse Restriktion als Gleichung einer Geraden auf (d.h. mit $=$).
2. Zeichne diese Gerade (z.B. Zweipunkt Methode).
3. Bestimme den **relevanten Halbraum** für die Restriktion.

Erklärung

Punkte im **relevanten Halbraum** erfüllen die ursprüngliche Restriktion.

Test

Wähle Punkt, der nicht auf der Geraden ist. Ist die Ungleichung erfüllt, so ist der Punkt im relevanten Halbraum, andernfalls außerhalb.

Schritt 1: Beispiel 1/2

Darstellung der Restriktion (I) $3x_1 + 2x_2 \leq 1200$

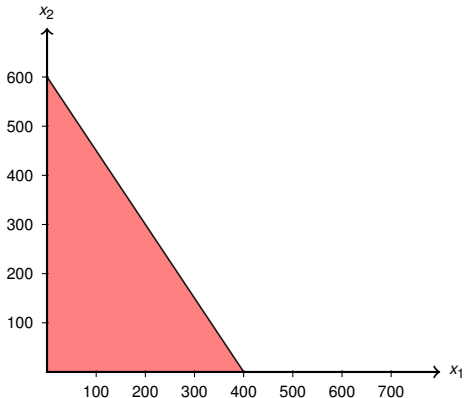
Zeichne Restriktion (I): $3x_1 + 2x_2 \leq 1200$

- 1. Fasse (I) als Gleichung auf:** $3x_1 + 2x_2 = 1200$
- 2. Zeichne Gerade:** Berechne z.B. zwei Punkte der Geraden:
 - ▶ Wähle $x_1 \stackrel{!}{=} 0 \Rightarrow x_2 = 600 \Rightarrow$ Punkt 1 ist $(0,600)$.
 - ▶ Wähle $x_2 \stackrel{!}{=} 0 \Rightarrow x_1 = 400 \Rightarrow$ Punkt 2 ist $(400,0)$.
- 3. Bestimme relevanten Halbraum:** Wähle einen beliebigen Punkt, z.B. $(0, 0)$.

$(0, 0)$ erfüllt (I), d.h. $3 \cdot 0 + 2 \cdot 0 \leq 1200$, daher liegt $(0, 0)$ im relevanten Halbraum.

Schritt 1: Beispiel 2/2

Darstellung der Restriktion (I) $3x_1 + 2x_2 \leq 1200$



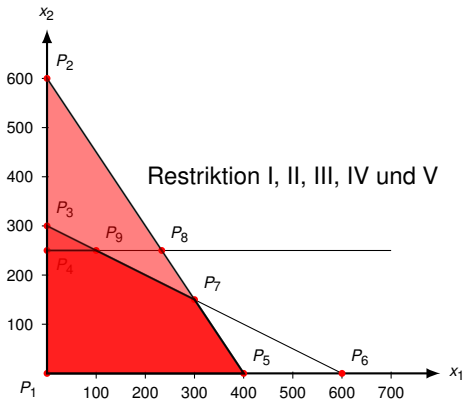
Problem

Es wird bislang nur eine Restriktion berücksichtigt.

Schritt 1: Menge der zulässigen Lösungen

Definition (Menge aller zulässigen Lösungen)

Wir bezeichnen die **Schnittmenge** aller relevanten Halbräume als die Menge M aller **zulässigen Lösungen** des LP.



Zeichne Isogewinngerade der Zielfunktion

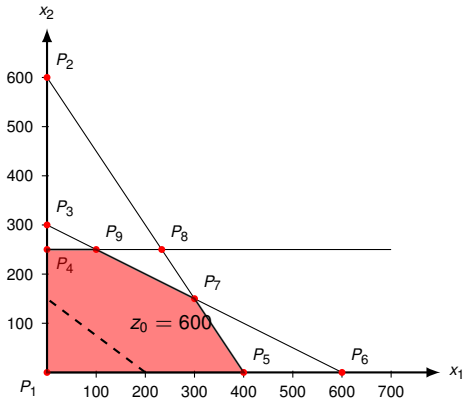
- ▶ **Ignoriere die Optimierungsvorschrift** (min / max)
- ▶ **Fasse Zielfunktion als Gleichung auf:** $z = c_1x_1 + c_2x_2$
- ▶ **Zeichne Isogewinngerade:** Wähle einen beliebigen Wert z_0 für z und setze ein

Beachte: Alle Punkte auf einer **Isogewinngeraden** haben den gleichen Zielfunktionswert.

Beispiel: Isogewinngerade für Zielfunktion $z = 3x_1 + 4x_2$

- ▶ **Wähle beliebigen Wert:** $z_0 = 600$.
- ▶ **Setze Gleichung:** $3x_1 + 4x_2 = 600$, d.h. alle Punkte haben Zielfunktionswert 600.
- ▶ **Bestimme Schnittpunkte:** x_1 -Achse (200, 0); x_2 -Achse (0, 150).

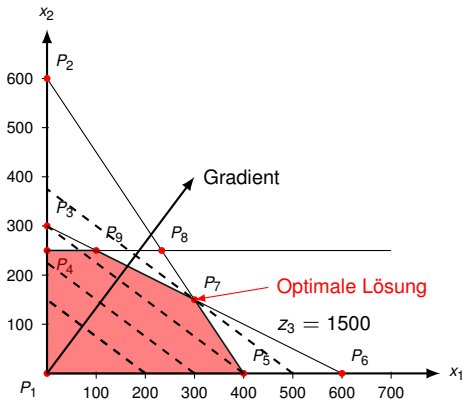
Beispiel: Isogewinngerade für Zielfunktion $z = 3x_1 + 4x_2$



Verschiebe Isogewinngerade

- ▶ **Bestimme die Steigung** der Zielfunktion (Gradient)
- ▶ **Verschiebe** die Isogewinngerade solange
 - ▶ in **Richtung** des Gradienten (**Maximierungsproblem**), andernfalls
 - ▶ in **negative Richtung** des Gradienten (**Minimierungsproblem**)
- ▶ bis die Isogewinngerade eine Lösung in einer **Ecke tangiert** und
- ▶ bei einer weiteren Verschiebung **keine zulässigen** Lösungen mehr auf der Isogewinngeraden liegen würden.

Beispiel: Isogewinnerade für Zielfunktion $z = 3x_1 + 4x_2$



Schritt 4: Ermittle Koordinaten einer optimalen Lösung

Beobachtung: Die gefundene optimale Lösung (Ecke) wird durch den Schnittpunkt (mindestens) zweier Restriktionsgeraden definiert.

Aktive Restriktion

Eine Restriktion ist für eine Lösung \mathbf{x} **aktiv**, wenn \mathbf{x} sie mit Gleichheit erfüllt.

Theorem

Die **optimale Lösung** \mathbf{x}^* ist durch das Gleichungssystem der **aktiven** Restriktionen gegeben.

Beispiel

Die Restriktionen (I) und (II) sind **aktiv**. Es folgt:

$$3x_1^* + 2x_2^* = 1200 \quad (\text{I})$$

$$5x_1^* + 10x_2^* = 3000 \quad (\text{II})$$

Auflösen ergibt $x_1^* = 300$ und $x_2^* = 150$.

Auswertung: Setze **optimale Lösung** \mathbf{x} in **Zielfunktion** $f(\mathbf{x})$ ein.

Beispiel

$$\mathbf{x}^* = (x_1^*, x_2^*) = (300, 150)$$

$$\text{Einsetzen: } z^* = f(x_1^*, x_2^*) = f(300, 150) = 3 \cdot 300 + 4 \cdot 150 = 1500$$

Ergebnis: Optimales Produktionsprogramm

Der maximale Deckungsbeitrag von $z^* = 1500$ GE wird bei einer Produktion von 300 Stück von Produkt P_1 und 150 Stück von Produkt P_2 erreicht.

Damit ist ein gewinnmaximales Produktionsprogramm gefunden.

Algorithmus

1. Bestimme die Menge \mathbb{M} aller **zulässigen Lösungen**.
2. Zeichne **eine** Isogewinngerade.
3. Verschiebe diese Isogewinngerade solange **parallel in Richtung des Gradienten** von $f(\mathbf{x})$ **bis durch weiteren Verschieb kein weiterer zulässiger Punkt** erreicht werden kann.
4. Berechne die Koordinaten von $\mathbf{x}^* = (x_1^*, x_2^*)$ durch **Auflösen des Gleichungssystems der aktiven Restriktionen**.
5. Berechne $z^* = f(x_1^*, x_2^*)$, z^* ist **der optimale Zielfunktionswert**.

Problemstellung:

- ▶ Ein Rind erhält täglich eine Mischung aus zwei Futtersorten S_1 und S_2 .
- ▶ Die Tagesration soll den Tagesbedarf der Nährstoffe I, II und III decken.

Restriktionen:

	Sorte S_1	Sorte S_2	Mindestbedarf
Nährstoff I	2 g/kg	1 g/kg	6 g
Nährstoff II	2 g/kg	4 g/kg	12 g
Nährstoff III	0 g/kg	4 g/kg	4 g
Preis	5 GE/kg	7 GE/kg	—

Frage

Welches Mischung ist bei Einhaltung der Nährstoffmenge kostenminimal?

Bedeutung des Mischungsproblems: Neben Landwirtschaft auch Farben/Chemie (BASF), Metallurgie; ähnliche

Problemformulierung (kein LP) durch amerikanische Wissenschaftler als Ernährung von Soldaten vor WW2. Später Nobelpreis.

1. Modellierung der Problemstellung als LP:

$$\min f(\mathbf{x}) = 5x_1 + 7x_2$$

$$2x_1 + x_2 \geq 6 \quad \text{Nährstoff I} \quad \text{(I)}$$

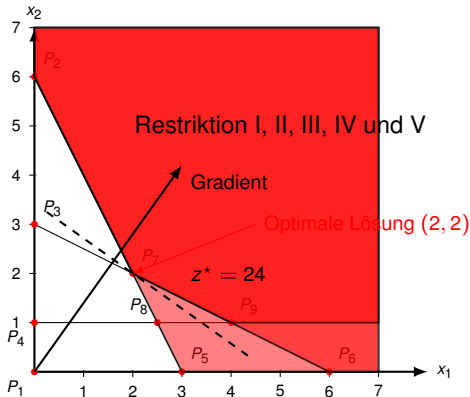
$$2x_1 + 4x_2 \geq 12 \quad \text{Nährstoff II} \quad \text{(II)}$$

$$4x_2 \geq 4 \quad \text{Nährstoff III} \quad \text{(III)}$$

$$x_1 \geq 0 \quad \text{(IV)}$$

$$x_2 \geq 0 \quad \text{(V)}$$

2. Welche Anzahl x_1 von S_1 und x_2 von S_2 soll verwendet werden?



Graphische Lösung von LP

Beispiel Produktionsprogrammplanung

Algorithmus

Beispiel

Eigenschaften zulässiger Lösungen

Normalform und Basis

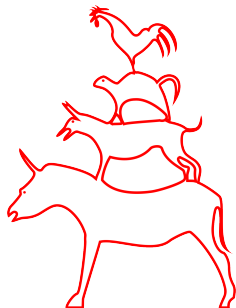
Standardform

Normalform

Basis

Zulässige Basislösung

Ecke



Linearkombination

Für Vektoren $x_i \in \mathbb{R}^n$ und Skalare $\lambda_i \in \mathbb{R}$, $i = 1, \dots, k$, bezeichnet

$$y = \sum_{i=1}^k \lambda_i x_i$$

eine **Linearkombination**.

Konvexe Linearkombination

Eine Linearkombination heißt **konvex**, wenn gilt

$$\lambda_i \geq 0 \quad \forall i = 1, \dots, k, \quad \text{und} \quad \sum_{i=1}^k \lambda_i = 1.$$

Bei einer **echten** konvexen Linearkombination gilt zudem $\lambda_i > 0$.

Theorem

Für jedes LP ist die Menge \mathbb{M} der zulässigen Lösungen **konvex**.

Ecke/Extrempunkt

Punkte einer konvexen Menge, die sich **nicht als echte** konvexe Linearkombination darstellen lassen, heißen **Ecke** bzw. **Extrempunkt**.

Satz – Eckensatz der linearen Optimierung

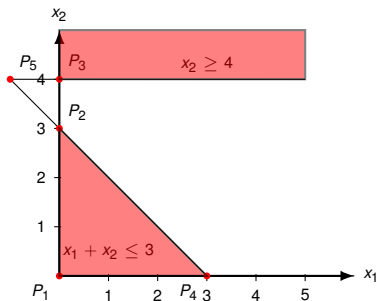
Wenn ein LP eine **optimale Lösung** besitzt, dann ist auch **mindestens** eine Ecke von \mathbb{M} eine optimale Lösung.

Sonderfall 1 – Es existiert keine zulässige Lösung

Leere zulässige Menge

Ist die Menge M der zulässigen Lösungen leer, d.h. $M = \emptyset$, so existiert **keine zulässige Lösung**.

Hintergrund: Die Restriktionen enthalten Widersprüche.

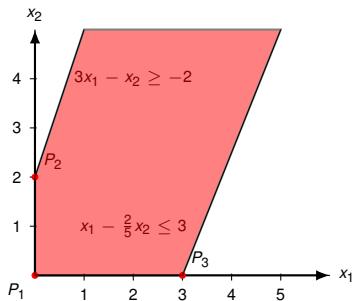


Praxis: Prüfen Modellierungsfehler? Alle Restriktionen wirklich nötig?

Unbeschränkter Lösungsraum

Ist die zulässige Menge nichtleer, d.h. $M \neq \emptyset$, und unbeschränkt bzgl. der Zielfunktion, d.h. $\sup z = \infty$ oder $\inf z = -\infty$, so existiert **keine optimale Lösung**.

Hintergrund: Die Restriktionen bilden kein Polyeder.

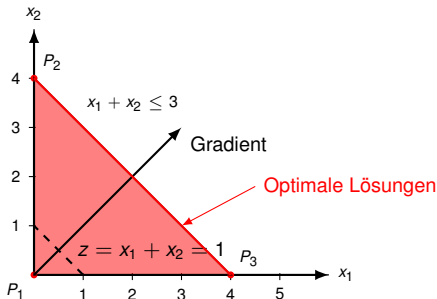


Praxis: Prüfen Modellierungsfehler? Restriktionen vergessen?

Duale Degeneriertheit

Das LP hat mehrere optimale Lösungen.

Hintergrund: Eine Restriktion ist parallel zur Isogewinngeraden.

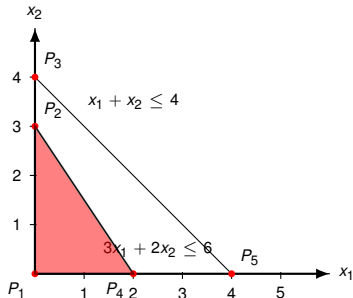


Praxis: kein Handlungsbedarf

Redundanz von Restriktionen

Mindestens eine Restriktion schneidet die zulässige Menge nicht.

Hintergrund: Mindestens eine Restriktion ist überflüssig.

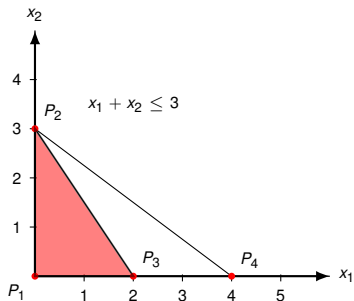


Praxis: Redundante Restriktion entfernen

Primale Degeneriertheit

Im n -dimensionalen Fall wird ein zulässiger Eckpunkt durch mindestens $n + 1$ Hyperebenen definiert.

Hintergrund: Mindestens eine Nebenbedingung ist überflüssig.



Praxis: kein Handlungsbedarf

Eckensatz der linearen Optimierung

Wenn ein LP eine optimale Lösung besitzt, dann ist auch **mindestens** eine Ecke von M eine optimale Lösung.

Konsequenz

Ein allgemeingültiges ($n > 2$) Lösungsverfahren muss v.a. prüfen, ob die **Sonderfälle 1 und 2** vorliegen und kann sich (im Wesentlichen) auf die Berechnung **von Ecken beschränken**.

Erläuterung

Ist die zulässige Menge weder leer noch unbeschränkt, dann existiert in mindestens einer Ecke (Schnittpunkt zweier Restriktionsgeraden) der konvexen Lösungsmenge eine Lösung.

Es kann sein, dass mehrere Lösungen optimal sind (Sonderfall 3) oder dieselbe Lösung durch mehrere Ecken beschrieben wird (Sonderfall 5).

Graphische Lösung von LP

Beispiel Produktionsprogrammplanung

Algorithmus

Beispiel

Eigenschaften zulässiger Lösungen

Normalform und Basis

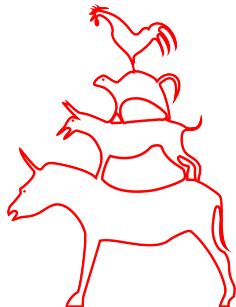
Standardform

Normalform

Basis

Zulässige Basislösung

Ecke



Lineares Programm in Standardform

Als lineares Programm in Standardform bezeichnen wir

$$\max f(\mathbf{x}) = c_1 x_1 + \dots + c_n x_n$$

$$\text{u.d.N.} \quad a_{i1} x_1 + \dots + a_{in} x_n \leq b_i, \quad \text{für } i = 1, \dots, m$$

$$x_j \geq 0, \quad \text{für } j = 1, \dots, n$$

Eigenschaften:

- ▶ Maximierungsproblem
- ▶ Nur Ungleichungsrestriktionen (kleiner gleich)
- ▶ Alle Entscheidungsvariablen sind vorzeichenbedingt

► Ungleichungen:

$$a_{i1}x_1 + \dots + a_{in}x_n \geq b_i \iff -a_{i1}x_1 - \dots - a_{in}x_n \leq -b_i$$

► Zielfunktion:

$$\min z = f(\mathbf{x}) \iff \max -z = -f(\mathbf{x})$$

► Gleichungen:

$$a_{i1}x_1 + \dots + a_{in}x_n = b_i \iff \begin{cases} a_{i1}x_1 + \dots + a_{in}x_n \geq b_i \\ a_{i1}x_1 + \dots + a_{in}x_n \leq b_i \end{cases}$$

► Nichtnegativität:

$$x_j \in \mathbb{R}$$

ist äquivalent zu den **zwei neuen Variablen** x' , x'' mit

$$x'_j \geq 0$$

$$x''_j \geq 0$$

$$x_j := x'_j - x''_j$$

Theorem

Jedes beliebige LP lässt sich in **Normalform** transformieren.

$$\begin{array}{rcll} \max Z = & c_1 x_1 + & \dots + & c_n x_n \\ \text{u.d.N.} & a_{11} x_1 + & \dots + & a_{in} x_n + & x_{n+1} & = & b_1 \\ & a_{21} x_1 + & \dots + & a_{2n} x_n + & & x_{n+2} & = & b_2 \\ & \vdots & \ddots & & & \ddots & \vdots \\ & a_{m1} x_1 + & \dots + & a_{mn} x_n + & & x_{n+m} & = & b_m \\ & x_1, & \dots, & x_n, & x_{n+1}, & \dots & x_{n+m} & \geq & 0 \end{array}$$

Schlupf

Eine Schlupfvariable x_{n+i} misst den **Schlupf** der i -ten-Restriktion

$$a_{i1}x_1 + \dots + a_{in}x_n + x_{n+i} = b_i \iff x_{n+i} = b_i - a_{i1}x_1 - \dots - a_{in}x_n$$

also den Umfang der nicht in Anspruch genommenen i -ten-Ressource.

Beispiel: Maschinenstunden des Produktionsprogramms

$$3x_1 + 2x_2 \leq 1200 \quad \text{Maschinen (h)} \quad (I)$$

$$3x_1 + 2x_2 + x_3 = 1200 \quad \text{Maschinen (h)} \quad (I')$$

Schlupfvariable misst die nicht verbrauchten Maschinenstunden.

Denomination:

- ▶ x_1, \dots, x_n nennt man **Strukturvariablen**.
- ▶ x_{n+1}, \dots, x_{n+m} nennt man **Schlupfvariablen**.

Basis

Jede nichtsinguläre $m \times m$ -Matrix **B** eines LP in Normalform heiSSt **Basis** und besteht aus den linear unabhängigen Spalten von **B**.

Basis-/Nichtbasisvariablen

Die m Spalten von **B** definieren die **Basisvariablen (BV)**.
Die restlichen n Variablen heiSSen **Nichtbasisvariablen (NBV)**.

Beispiel – (I'), (II'), (III') des Produktionsprogramms

$$x_3 = 1200 - 3x_1 - 2x_2$$

$$x_4 = 3000 - 5x_1 - 10x_2 \quad \implies \text{BV: } x_3, x_4, x_5, \text{ NBV: } x_1, x_2$$

$$x_5 = 125 - 0.5x_2$$

Basislösung

Setze NBV auf 0. Dann nennen wir die erhaltene eindeutige Lösung für eine Basis **B** **Basislösung**.

Beispiel (Fortsetzung)

NBV $x_1 = 0, x_2 = 0 \implies$ Basislösung $\mathbf{x} = (0, 0, 1200, 3000, 125)$.

Zulässigkeit und Degeneriertheit

Eine Basislösung ist

- ▶ **zulässig**, falls alle BV der Basislösung nichtnegative Werte annehmen, sonst unzulässig;
- ▶ **degeneriert**, falls mindestens eine der BV in der Basislösung den Wert 0 annimmt.

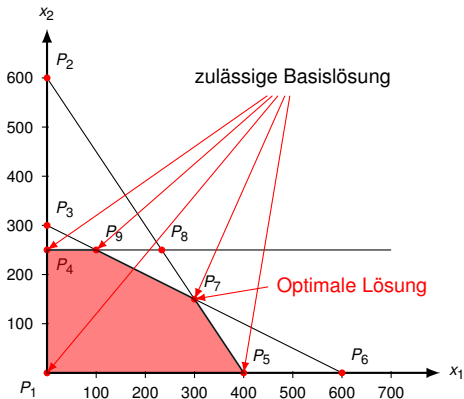
Beispiel (Fortsetzung)

Basislösung $\mathbf{x} = (0, 0, 1200, 3000, 125)$ ist zulässig und nicht-degeneriert.

Für jede Ecke gibt es eine zulässige Basislösung

Theorem

Für ein LP in Normalform sind die Ecken von \mathbb{M} genau die **zulässigen Basislösungen**.



Fragen zur Wiederholung:

- ▶ Wie kann ein LP mit zwei Variablen **graphische** gelöst werden?
- ▶ Wie kann ein beliebiges LP in **Normalform transformiert** werden?
- ▶ Wozu sind **Schlupfvariablen** zu ergänzen?

Klausurrelevante Literatur

Domschke/Drexel: Kapitel 2.2 und 2.3, oder Nickel/Stein/Waldman Kapitel 1.4 und 1.5, oder Werner Kapitel 2.1 und 2.2.2, oder Suhl/Mellouli Kapitel 2.2 und 2.3

Ausblick

- ▶ Tutorium: Graphische Lösung & Transformation
- ▶ Simplex-Verfahren: Domschke/Drexel Kapitel 2.4, oder Werner Kapitel 2.2 und 3.1, oder Nickel/Stein/Waldman Kapitel 1.6 und 1.7, oder Suhl/Mellouli Kapitel 2.6

Teil III

Lineare Optimierung II – Primales und Duales Simplex Verfahren

Problemstellung als LP:

$$\max z = f(\mathbf{x}) = 3x_1 + 4x_2$$

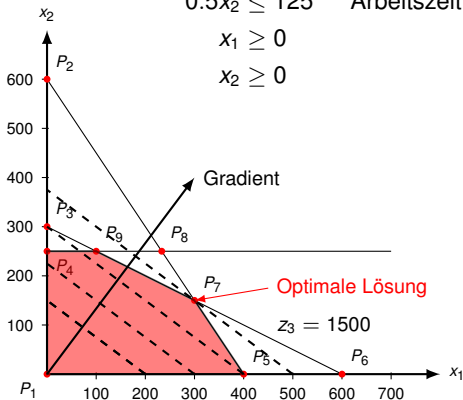
$$3x_1 + 2x_2 \leq 1200 \quad \text{Maschinen (h)} \quad \text{(I)}$$

$$5x_1 + 10x_2 \leq 3000 \quad \text{Rohstoffe (ME)} \quad \text{(II)}$$

$$0.5x_2 \leq 125 \quad \text{Arbeitszeit (h)} \quad \text{(III)}$$

$$x_1 \geq 0 \quad \text{(IV)}$$

$$x_2 \geq 0 \quad \text{(V)}$$



65 Wiederholung

Primales
Simplex-Verfahrens

Das Simplex-Tableau

Der primale
Simplex-AlgorithmusDer duale
Simplex-AlgorithmusZusammenfassung &
Ausblick

Mögliche Sonderfälle:

1. Keine zulässige Lösung, d.h. $M = \emptyset$
2. Unbeschränkter Lösungsraum (mit Blick auf Zielfunktion)
3. Duale Degeneriertheit – mehrere optimale Lösungen
4. Redundante Restriktionen
5. Primale Degeneriertheit – Ecke durch mehr Restriktionen als nötig definiert

Theorem

Jedes beliebige LP lässt sich durch Transformationsvorschriften in Normalform mit n Strukturvariablen, m Schlupfvariablen sowie m Restriktionen schreiben.

$$\begin{array}{rcll}
 \max z = & c_1 x_1 + & \dots + & c_n x_n \\
 \text{u.d.N.} & a_{11} x_1 + & \dots + & a_{in} x_n + & x_{n+1} & = & b_1 \\
 & a_{21} x_1 + & \dots + & a_{2n} x_n + & & x_{n+2} & = & b_2 \\
 & \vdots & \ddots & & & \ddots & \vdots \\
 & a_{m1} x_1 + & \dots + & a_{mn} x_n + & & x_{n+m} & = & b_m \\
 & x_1, & \dots, & x_n, & x_{n+1}, & \dots & x_{n+m} & \geq & 0
 \end{array}$$

- Primales Simplex-Verfahren
- Das Simplex-Tableau
- Der primale Simplex-Algorithmus
- Der duale Simplex-Algorithmus
- Zusammenfassung & Ausblick

Primales
Simplex-Verfahrens

Das Simplex-Tableau

Der primale
Simplex-Algorithmus

Der duale
Simplex-Algorithmus

Zusammenfassung &
Ausblick

LP aus der Praxis haben häufig sehr viele
Entscheidungsvariablen.



**Wie können wir LPs mit mehr als zwei
Entscheidungsvariablen lösen?**

Primales Simplex-Verfahren

Basislösung und Ecke

Naiver Ansatz zur Berechnung einer optimalen Lösung

Beispiel für Arbeitsweise des Simplex-Verfahrens

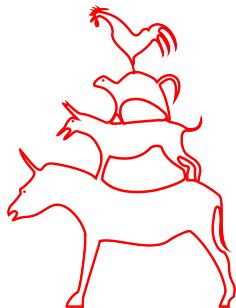
Das Simplex-Tableau

Der primale Simplex-Algorithmus

Vollständiges Beispiel

Der duale Simplex-Algorithmus

Vollständiges Beispiel



68 **Primales Simplex-Verfahrens**

Basislösung und Ecke
Naiver Ansatz zur Berechnung einer optimalen Lösung
Beispiel für Arbeitsweise des Simplex-Verfahrens

Das Simplex-Tableau

Der primale Simplex-Algorithmus

Der duale Simplex-Algorithmus

Zusammenfassung & Ausblick

Optimale Lösung

Wenn eine **optimale Lösung** existiert, dann liegt auch in **mindestens einer Ecke** des Lösungsraums \mathbb{M} eine optimale Lösung.

Basis-/Nichtbasisvariablen

Das Gleichungssystem eines LP in Normalform besteht aus n **Strukturvariablen** und m **Schlupfvariablen**, aber lediglich aus m Gleichungen.

Ecken und zulässige Basislösungen

Für ein LP in Normalform sind die **Ecken** von \mathbb{M} genau die **zulässigen Basislösungen**.

69 **Primales Simplex-Verfahrens**

- Basislösung und Ecke
- Naiver Ansatz zur Berechnung einer optimalen Lösung
- Beispiel für Arbeitsweise des Simplex-Verfahrens

Das Simplex-Tableau

Der primale Simplex-Algorithmus

Der duale Simplex-Algorithmus

Zusammenfassung & Ausblick

Basis

Jede nichtsinguläre $m \times m$ -Matrix **B** eines LP in Normalform heiSSt **Basis** und besteht aus den linear unabhängigen Spalten von **B**.

Basis-/Nichtbasisvariablen

Die m Spalten von **B** definieren die **Basisvariablen (BV)**.
Die restlichen n Variablen heiSsen **Nichtbasisvariablen (NBV)**.

Basislösung

Setze NBV auf 0. Dann nennen wir die erhaltene eindeutige Lösung für eine Basis **B** **Basislösung**.

Zulässige Basislösung

Eine Basislösung ist **zulässig**, falls alle BV der Basislösung nichtnegative Werte annehmen, sonst unzulässig.

Operations Research

J. Pannek

Wiederholung

Primales
Simplex-Verfahrens

70 Basislösung und Ecke

Naiver Ansatz zur
Berechnung einer
optimalen Lösung

Beispiel für Arbeitsweise
des Simplex-Verfahrens

Das Simplex-Tableau

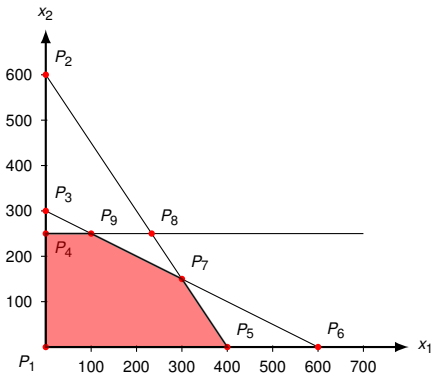
Der primale
Simplex-Algorithmus

Der duale
Simplex-Algorithmus

Zusammenfassung &
Ausblick

Wie gelangt man von einer Ecke zur Basislösungen?

$$\begin{aligned} \max \quad & z = 3x_1 + 4x_2 \\ \text{s.t.} \quad & 3x_1 + 2x_2 + x_3 = 1200 \\ & 5x_1 + 10x_2 + x_4 = 3000 \\ & 0.5x_2 + x_5 = 125 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$



Frage: Wie lautet die Basislösung zu P_9 ?

Ablezen: $x_1 = 100, x_2 = 250$

Einsetzen: $x_3 = 400, x_4 = x_5 = 0$

\Rightarrow Nichtbasisvariablen x_4, x_5

Basisvariablen x_1, x_2, x_3

Alle BV sind positiv, d.h. die Basislösung ist **zulässig**.

Problem

Ein LP in Normalform hat $n + m$ Variablen und m Restriktionen, n Variablen sind **nicht erklärt**.

Idee

1. Setze n der $n + m$ Variablen x_i gleich null.
2. Löse (falls möglich) das Gleichungssystem.

Produktionsprogrammplan vorherige Folie

Es gilt $n = 2$, $m = 3$. Setze z.B. x_2 und x_4 auf den Wert 0.

Gleichungssystem auflösen ergibt $x_1 = 600$, $x_5 = 125$, $x_3 = -600$.

Die Basislösung ist unzulässig ($x_3 < 0$) und entspricht dem Punkt P_6 .

Ein naiver Ansatz zur Berechnung einer optimalen Basislösung 1/2

Naiver Ansatz

1. Setze n der $n + m$ Variablen x_i gleich null.
2. Löse (falls möglich) das Gleichungssystem.
3. Berechne für jede Basislösung $f(\mathbf{x})$.
4. Wähle ein optimales \mathbf{x} aus.

Problem:

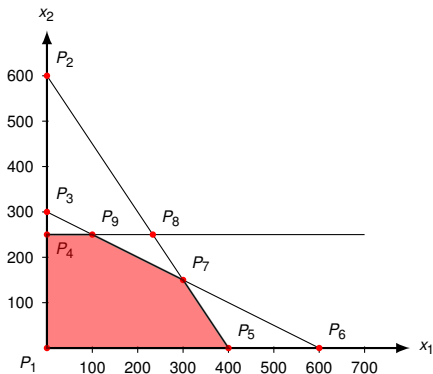
Es sind $\binom{n+m}{n}$ mögliche Basislösungen zu berechnen, für die meisten Praxisprobleme ist das viel zu aufwendig.

Onlinerechner für Binomialkoeffizienten

Ein naiver Ansatz zur Berechnung einer optimalen Basislösung 2/2

Gegeben $n = 2$, $m = 3$: Dann existieren $\binom{2+3}{2} = 10$ mögliche Basislösungen.

NBV	BV	$z = f(\mathbf{x})$
$x_1 = x_2 = 0$:	$x_3 = 1200, x_4 = 3000, x_5 = 125$	0
$x_1 = x_3 = 0$:	$x_2 = 600, x_4 = -3000, x_5 = -175$	—
$x_1 = x_4 = 0$:	$x_2 = 300, x_3 = 600, x_5 = -25$	—
$x_1 = x_5 = 0$:	$x_2 = 250, x_3 = 700, x_4 = 500$	1000
$x_2 = x_3 = 0$:	$x_1 = 400, x_4 = 1000, x_5 = 125$	1200
$x_2 = x_4 = 0$:	$x_1 = 600, x_3 = -600, x_5 = 125$	—
$x_2 = x_5 = 0$:	nicht lösbar, da a^1, a^3, a^4 lin. abhängig	—
$x_3 = x_4 = 0$:	$x_1 = 300, x_2 = 150, x_5 = 50$	1500
$x_3 = x_5 = 0$:	$x_1 = 233 \frac{1}{3}, x_2 = 250, x_4 = -666 \frac{2}{3}$	—
$x_4 = x_5 = 0$:	$x_1 = 100, x_2 = 250, x_3 = 400$	1300



Produktionsprogrammplanung – LP in Normalform:

$$\begin{aligned} \max z &= 3x_1 + 4x_2 + 0x_3 + 0x_4 + 0x_5 \\ 3x_1 + 2x_2 + x_3 &= 1200 & \text{(I')} \\ 5x_1 + 10x_2 + x_4 &= 3000 & \text{(II')} \\ 0.5x_2 + x_5 &= 125 & \text{(III')} \\ x_1, x_2, x_3, x_4, x_5 &\geq 0 & \text{(IV')} \end{aligned}$$

Startpunkt:

$\mathbf{x}^1 = (0, 0, 1200, 3000, 125)$ ist zulässige Basislösung mit NBV

$x_1 = x_2 = 0$ und BV $x_3 = 1200, x_4 = 3000, x_5 = 125$.

- ▶ Eine Erhöhung der Produktionsmenge x_2 um 1 führt zu einem **Anstieg des Zielfunktionswerts z um 4 GE (stärkster Anstieg)**.
- ▶ Basierend auf \mathbf{x}^1 kann die NBV x_2 um **höchstens 250 Stück erhöht** werden, andernfalls wird die **Restriktion (III') verletzt**. (I') und (II') werden bei $x_2 = 250$ nicht verletzt.
- ▶ x_5 wird von 125 **auf 0 vermindert**, damit (III') eingehalten wird.

- ▶ Der Wert von x_2 wird erhöht, der Wert von x_5 wird vermindert, d.h. es erfolgt ein **Basistausch**, x_5 wird neue NBV, x_2 wird neue BV.
- ▶ Durch diesen Tausch gelangt man zu einer **neuen Basislösung** bzw. zu einer **neuen Ecke**.
- ▶ **Durchführung des Basistausches:**
 1. (III') $\frac{1}{2}x_2 + x_5 = 125$ wird nach $x_2 = 250 - 2x_5$ umgeformt
 2. Ergebnis wird ins Gleichungssystem z, (I)-(III) eingesetzt:

$$\begin{array}{rcll} \max z = 3x_1 & & -8x_5 + 1000 & \\ & 3x_1 & + x_3 & -4x_5 = 700 & \text{(I''')} \\ & 5x_1 & & + x_4 - 20x_5 = 500 & \text{(II''')} \\ & & x_2 & + 2x_5 = 250 & \text{(III''')} \end{array}$$

Neue Basislösung: $x^2 = (0, 250, 700, 500, 0)$ mit $z = 1000$.

- ▶ Eine Erhöhung der Produktionsmenge x_1 um 1 führt zu einem **Anstieg des Zielfunktionswerts** z um 3 GE (*stärkster Anstieg*).
- ▶ Basierend auf \mathbf{x}^2 kann die NBV x_1 um **höchstens 100 Stück erhöht** werden, andernfalls wird die **Restriktion (II'') verletzt**. (I'') würde erst bei $x_2 \geq 233,33$ verletzt werden.
- ▶ x_4 wird von 500 **auf 0 vermindert**, damit (II'') eingehalten wird.
- ▶ **Durchführung des Basistausches:**
 1. (II'') $5x_1 + x_4 - 20x_5 = 500$ wird nach $x_1 = 100 - \frac{1}{5}x_4 + 4x_5$ umgeformt
 2. Ergebnis wird ins Gleichungssystem z , (I'')-(III'') eingesetzt:

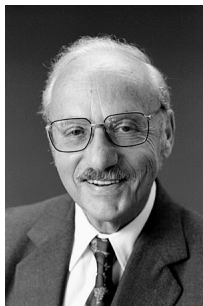
$$\begin{array}{rcl} \max z = & -3/5x_4 + 4x_5 + 1300 & \\ & x_3 - 3/5x_4 + 8x_5 = 400 & \text{(I'')} \\ x_1 & + 1/5x_4 - 4x_5 = 100 & \text{(II'')} \\ x_2 & + 2x_5 = 250 & \text{(III'')} \end{array}$$

Neue Basislösung: $\mathbf{x}^3 = (100, 250, 400, 0, 0)$ mit $z = 1300$.

- ▶ x_5 kann um 50 **erhöht** werden (*stärkster Anstieg*), wenn x_3 **vermindert** wird.
- ▶ **Durchführung des Basistausches:**
 1. (I'') $x_3 - 3/5x_4 + 8x_5 = 400$ wird nach $x_5 = 50 - \frac{1}{8}x_3 + \frac{3}{40}x_4$ umgeformt
 2. Ergebnis wird ins Gleichungssystem z, (I''')-(III''') eingesetzt:

$$\begin{array}{rcll} z = & -1/2x_3 + 3/10x_4 & +1500 & \\ & 1/8x_3 - 3/40x_4 + x_5 & = 50 & \text{(I''')} \\ x_1 & +1/2x_3 - 1/10x_4 & = 300 & \text{(II''')} \\ & x_2 - 1/4x_3 + 3/20x_4 & = 150 & \text{(III''')} \end{array}$$

Neue Basislösung: $x^4 = (300, 150, 0, 0, 50)$ mit $z = 1500$.



George Dantzig¹
1914 – 2005

Der Simplex-Algorithmus zählt zu den zehn einflussreichsten mathematischen Verfahren des 20. Jahrhunderts.

Operations Research

J. Pannek

Wiederholung

Primales
Simplex-Verfahrens

Basislösung und Ecke

Naiver Ansatz zur
Berechnung einer
optimalen Lösung

80 Beispiel für Arbeitsweise
des Simplex-Verfahrens

Das Simplex-Tableau

Der primale
Simplex-Algorithmus

Der duale
Simplex-Algorithmus

Zusammenfassung &
Ausblick

¹amerikanische Mathematiker/Informatiker/Physiker

Primales Simplex-Verfahrens

Basislösung und Ecke

Naiver Ansatz zur Berechnung einer optimalen Lösung

Beispiel für Arbeitsweise des Simplex-Verfahrens

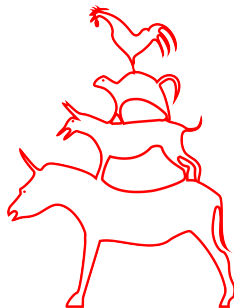
Das Simplex-Tableau

Der primale Simplex-Algorithmus

Vollständiges Beispiel

Der duale Simplex-Algorithmus

Vollständiges Beispiel



Darstellung eines LP mit Hilfe eines Simplex-Tableaus 1/2

Gegeben: LP in Normalform mit Schlupfvariablen x_3, x_4, x_5 :

$$\begin{aligned} \max z &= 3x_1 + 4x_2 + 0x_3 + 0x_4 + 0x_5 \\ 3x_1 + 2x_2 + x_3 &= 1200 && \text{(I')} \\ 5x_1 + 10x_2 + x_4 &= 3000 && \text{(II')} \\ 0.5x_2 + x_5 &= 125 && \text{(III')} \\ x_1, x_2, x_3, x_4, x_5 &\geq 0 && \text{(IV')} \end{aligned}$$

BV	Strukturvariablen		Schlupfvariablen			RS
	x_1	x_2	x_3	x_4	x_5	
x_3	3	2	1	0	0	1200
x_4	5	10	0	1	0	3000
x_5	0	0.5	0	0	1	125
z	-3	-4	0	0	0	0

Zielfunktion umformen: $z = 3x_1 + 4x_2 \rightarrow z - 3x_1 - 4x_2 = 0$

Basisvariablen bei Position der 1er in Einheitsvektoren

BV	Strukturvariablen				Schlupfvariablen				
	x_1	x_2	\dots	x_n	x_{n+1}	x_{n+2}	\dots	x_{n+m}	
x_{n+1}	a_{11}	a_{12}	\dots	a_{1n}	1	0	\dots	0	b_1
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots
x_{n+m}	a_{m1}	a_{m2}	\dots	a_{mn}	0	0	\dots	1	b_m
Z	$-c_1$	$-c_2$	\dots	$-c_n$	0	0	\dots	0	0

Generierung des Tableaus

- ▶ BV-Spalte: Variablen, die im aktuellen Tableau die Basisvariablen bilden, definiert über Einheitsvektoren
- ▶ Zelle rechts unten: Zielfunktionswert der aktuellen Basislösung
- ▶ z-Zeile: Umformung der Zielfunktion zu $z - c_1 x_1 - \dots - c_n x_n = 0$.
- ▶ **Achtung:** Es gilt $x_i \geq 0$; wird nicht in Tableau vermerkt.

Beantworten Sie unten stehenden Fragen für das gegebene Tableau:

BV	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RS
x_3	0	2	1	0	0	-20	4	35
x_1	1	7	0	0	0	1	12	240
x_4	0	-2	0	1	0	6	78	20
x_5	0	4	0	0	1	2	5	90
z	5	-3	3	0	0	-6	0	56

- ▶ Welchen Wert hat die Zielfunktion?
- ▶ Ist x_1 eine Basisvariable?
- ▶ Ist x_2 eine Basisvariable?
- ▶ Welchen Wert haben die Variablen x_2 und x_4 ?
- ▶ Wie heißen die Schlupfvariablen?
- ▶ Wie lautet die Zielfunktion?
- ▶ Wie verändert sich der z-Wert, wenn der Wert von x_2 um eins erhöht wird?
- ▶ Wie verändert sich der z-Wert, wenn der Wert von x_1 um eins erhöht wird?
- ▶ Bei Erhöhung welchen Variablenwertes ändert sich der z-Wert am stärksten?

Primales Simplex-Verfahrens

Basislösung und Ecke

Naiver Ansatz zur Berechnung einer optimalen Lösung

Beispiel für Arbeitsweise des Simplex-Verfahrens

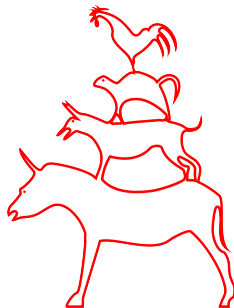
Das Simplex-Tableau

Der primale Simplex-Algorithmus

Vollständiges Beispiel

Der duale Simplex-Algorithmus

Vollständiges Beispiel



Algorithmus 1: Primaler Simplex-Algorithmus

Input : zulässige Basislösung für ein LP in Normalform

1 Stelle Anfangstableau auf

2 **while** Abbruchbedingung (I oder II) nicht erfüllt **do**

3 | **Schritt 1:** Wähle **Pivotspalte** t .

4 | **Schritt 2:** Wähle **Pivotzeile** s .

5 | **Schritt 3:** Vollziehe **Basistausch**, d.h. erzeuge einen Einheitsvektor in
 | der Spalte t , sodass $a'_{st} = 1$.

6 **end**

Output: optimaler Zielfunktionswert z^* sowie *eine* optimale Lösung x^*
oder Meldung über auf \mathbb{M} unbeschränkter Zielfunktion

Primale Abbruchbedingung I

Falls in z -Zeile **kein negativer Eintrag** existiert, dann ist die optimale Lösung gefunden und **das Verfahren endet!**

Primale Auswahlregel für die Pivotspalte t

Betrachte **z -Zeile** und wähle die **Spalte t** in welcher der **kleinste negative** Koeffizient steht.

(Falls es mehrere Spalten mit kleinstem negativen Koeffizienten gibt, wähle davon eine beliebige Spalte.)

- ▶ Die Spalte t heißt **Pivotspalte**.
- ▶ Die Variable x_t ist NBV und soll in die Basis aufgenommen werden.

Primale Abbruchbedingung II

Sind in der Pivotspalte t alle Koeffizienten **null oder negativ** ($a_{it} \leq 0, i = 1, \dots, m$), dann **endet das Verfahren**. Der Lösungsraum ist in diesem Fall unbeschränkt (Sonderfall 2).

Primale Auswahlregel für die Pivotzeile s

Betrachte alle **positiven** Koeffizienten a_{it} der Pivotspalte t :
Wähle nun die **Zeile** s mit

$$\frac{b_s}{a_{st}} = \min \left\{ \frac{b_i}{a_{it}} \mid i = 1, \dots, m \text{ und } a_{it} > 0 \right\}$$

- ▶ Die Zeile s heit **Pivotzeile**. x_s soll die Basis verlassen.
- ▶ Das gemeinsame Element a_{st} der Pivotzeile s und Pivotspalte t heit **Pivotelement**.

- ▶ Schritt 1 besagt, dass x_t neue BV werden soll.
Schritt 2 besagt, dass dafür x_s die Basis verlassen soll und NBV wird.
- ▶ Dies nennt man **Basistausch**.
- ▶ Zur Umsetzung des Basistausch ist in der Pivotspalte t ein **Einheitsvektor** mit $a_{st} = 1$ zu erzeugen.
- ▶ Dazu verwendet man das **GauSS-Jordan-Eliminationsverfahren** (alternativ: *Dreiecksregel*).
 1. Dividiere alle Werte der Pivotzeile durch a_{st} .
Nun steht an der Position des Pivotelements eine 1.
 2. Addiere das a_{it} -fache der neuen Pivotzeile zur Zeile i ($\forall i, i \neq s$).

Gegebenes LP, liegt bereits in Normalform vor

$$\begin{aligned}\max z &= 3x_1 + 4x_2 + 0x_3 + 0x_4 + 0x_5 \\ 3x_1 + 2x_2 + x_3 &= 1200 && \text{(I')} \\ 5x_1 + 10x_2 + x_4 &= 3000 && \text{(II')} \\ 0.5x_2 + x_5 &= 125 && \text{(III')} \\ x_1, x_2, x_3, x_4, x_5 &\geq 0 && \text{(IV')}\end{aligned}$$

Initiales Tableau aufstellen (Algorithmus 1, Zeile 1):

BV	x_1	x_2	x_3	x_4	x_5	RS
x_3	3	2	1	0	0	1200
x_4	5	10	0	1	0	3000
x_5	0	$\frac{1}{2}$	0	0	1	125
z	-3	-4	0	0	0	0

Tableau vor 1. Iteration:

BV	x_1	x_2	x_3	x_4	x_5	RS	$\frac{b_i}{a_{if}}$
x_3	3	2	1	0	0	1200	$\frac{1200}{2} = 600$
x_4	5	10	0	1	0	3000	$\frac{3000}{10} = 300$
x_5	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	125	$\frac{125}{1/2} = 250$
z	-3	-4	0	0	0	0	

Tableau nach 1. Iteration:

	BV	x_1	x_2	x_3	x_4	x_5	RS	
(I')	x_3	3	0	1	0	-4	700	(I) - 2 · (III')
(II')	x_4	5	0	0	1	-20	500	(II) - 10 · (III')
(III')	x_2	0	1	0	0	2	250	2 · (III)
(IV')	z	-3	0	0	0	8	1000	(IV) + 4 · (III')

Tableau vor 2. Iteration:

BV	x_1	x_2	x_3	x_4	x_5	RS	$\frac{b_i}{a_{it}}$
x_3	3	0	1	0	-4	700	$\frac{700}{3} = 233\frac{1}{3}$
x_4	55	0	0	1	-20	500	$\frac{500}{5} = 100$
x_2	0	1	0	0	2	250	$\frac{250}{0} = \infty$
z	-3	-3	0	0	8	1000	

Tableau nach 2. Iteration:

	BV	x_1	x_2	x_3	x_4	x_5	RS	
(I'')	x_3	0	0	1	$-\frac{3}{5}$	8	400	(I') - 3 · (II'')
(II'')	x_1	1	0	0	$\frac{1}{5}$	-4	100	(II') · $\frac{1}{5}$
(III'')	x_2	0	1	0	0	2	250	(III') - 0 · (II'')
(IV'')	z	0	0	0	$\frac{3}{5}$	-4	1300	(IV') + 3 · (II'')

Tableau vor 3. Iteration:

BV	x_1	x_2	x_3	x_4	x_5	RS	$\frac{b_i}{a_{it}}$
x_3	0	0	1	$-\frac{3}{5}$	88	400	$\frac{400}{8} = 50$
x_1	1	0	0	$\frac{1}{5}$	-4	100	—
x_2	0	1	0	0	2	250	$\frac{250}{2} = 125$
z	0	0	0	$\frac{3}{5}$	-4-4	1300	

Tableau nach 3. Iteration:

	BV	x_1	x_2	x_3	x_4	x_5	RS	
(I''')	x_5	0	0	$\frac{1}{8}$	$-\frac{3}{40}$	1	50	(I'') $\cdot \frac{1}{8}$
(II''')	x_1	1	0	$\frac{1}{2}$	$-\frac{1}{10}$	0	300	(II'') + 4 \cdot (I''')
(III''')	x_2	0	1	$-\frac{1}{4}$	$\frac{3}{20}$	0	150	(III'') - 2 \cdot (I''')
(IV''')	z	0	0	$\frac{1}{2}$	$\frac{3}{10}$	0	1500	(IV'') + 4 \cdot (I''')

Tableau vor 4. Iteration:

BV	x_1	x_2	x_3	x_4	x_5	RS
x_5	0	0	$\frac{1}{8}$	$-\frac{3}{40}$	1	50
x_1	1	0	$\frac{1}{2}$	$-\frac{1}{10}$	0	300
x_2	0	1	$-\frac{1}{4}$	$\frac{3}{20}$	0	150
z	0	0	$\frac{1}{2}$	$\frac{3}{10}$	0	1500

Alle $c_i \geq 0 \implies$ **Primale Abbruchbedingung I erfüllt**
 \implies Der Zielfunktionswert lässt sich nicht mehr steigern.

Optimale Lösung aus 4. Tableau ablesen

- ▶ $\mathbf{x}^* = (300, 150, 0, 0, 50)$ mit einem maximalen Gewinn von 1500 GE
- ▶ Produktion: 300 P_1 , 150 P_2
- ▶ Rest: 50 Arbeitsstunden

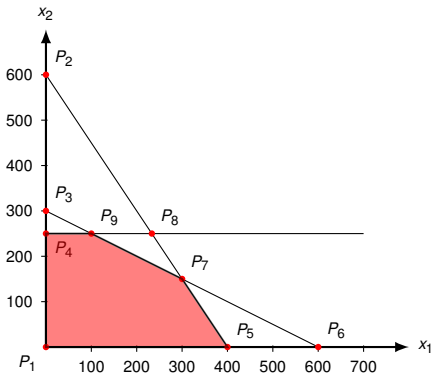
Grafische Veranschaulichung des Lösungsfortschritts

Tableau 1 – BV: $x_3 = 1200, x_4 = 3000, x_5 = 125 \rightarrow P_1, z = 0$

Tableau 2 – BV: $x_2 = 250, x_3 = 700, x_4 = 500 \rightarrow P_4, z = 1000$

Tableau 3 – BV: $x_1 = 100, x_2 = 250, x_3 = 400 \rightarrow P_9, z = 1300$

Tableau 4 – BV: $x_1^* = 300, x_2^* = 150, x_5^* = 50 \rightarrow P_7, z^* = 1500$



Operations Research

J. Pannek

Wiederholung

Primales
Simplex-Verfahrens

Das Simplex-Tableau

Der primale
Simplex-Algorithmus

93 Vollständiges Beispiel

Der duale
Simplex-Algorithmus

Zusammenfassung &
Ausblick

Primales Simplex-Verfahrens

Basislösung und Ecke

Naiver Ansatz zur Berechnung einer optimalen Lösung

Beispiel für Arbeitsweise des Simplex-Verfahrens

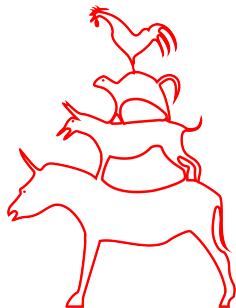
Das Simplex-Tableau

Der primale Simplex-Algorithmus

Vollständiges Beispiel

Der duale Simplex-Algorithmus

Vollständiges Beispiel



Primales Simplex-Verfahren erfordert *zulässige*
Basislösung



Berechnung mit dem dualen Simplex-Verfahren

Algorithmus 2: Dualer Simplex-Algorithmus

Input : unzulässige Basislösung für ein LP in Normalform

- 1 Stelle Anfangstableau auf
- 2 **while** *Duale Abbruchbedingung (I oder II) nicht erfüllt* **do**
 - 3 | **Schritt 1:** Wähle **Pivotzeile** s .
 - 4 | **Schritt 2:** Wähle **Pivotspalte** t .
 - 5 | **Schritt 3:** Vollziehe Basistausch, d.h. erzeuge Einheitsvektor in Spalte t , sodass $a'_{st} = 1$.
- 6 **end**

Output: zulässige Basislösung **oder** Meldung über unlösbares Problem wg. $M = \emptyset$

Beachte: Primaler Simplex bestimmt erst Pivotspalte, dann Pivotzeile.

Duale Abbruchbedingung I

Falls alle b'_i der RS positiv sind ($b'_i \geq 0, i = 1, \dots, m$), so liegt bereits eine zulässige Basislösung vor. **Das Verfahren endet.**

Duale Auswahlregel Pivotzeile

Wähle Zeile s mit kleinstem negativen $b'_i < 0, i = 1, \dots, m$.

(Falls es mehrere Zeilen mit kleinstem negativen Koeffizienten gibt, wähle davon eine beliebige Zeile.)

- ▶ Die Zeile s heißt **Pivotzeile**. x_s soll die Basis verlassen.

Duale Abbruchbedingung II

Sind alle Koeffizienten der Pivotzeile positiv ($a'_{sj} \geq 0, \forall j$), so besitzt das Problem keine zulässige Basislösung (Sonderfall 1, $\mathbb{M} = \emptyset$).

Abbruch des gesamten Simplex-Verfahrens!

Duale Auswahlregel Pivotspalte

Betrachte in der der Pivotzeile s alle Spalten mit **negativen** Koeffizienten a'_{sj} ($j = 1, \dots, j$):

Wähle eine Spalte t mit

$$\frac{c'_t}{a'_{st}} = \max \left\{ \frac{c'_j}{a'_{sj}} \mid \forall j \text{ und } a'_{st} < 0 \right\}.$$

- Die Spalte t heißt **Pivotspalte**.

Identisch mit Vorgehen des primalen Simplex-Verfahrens.

Gegebenes LP:

$$\max z = 2x_1 + x_2$$

$$x_1 + x_2 \geq 8 \quad \text{(I)}$$

$$3x_1 + x_2 \geq 12 \quad \text{(II)}$$

$$x_1 + x_2 \leq 10 \quad \text{(III)}$$

$$x_1, x_2 \geq 0$$

Transformation in Normalform:

- ▶ Multiplikation von (I) und (II) mit (-1)
- ▶ Hinzunahme von drei Schlupfvariablen x_3, x_4, x_5

$$\begin{aligned} \max z &= 2x_1 + x_2 \\ -x_1 - x_2 + x_3 &= -8 && \text{(I)} \\ -3x_1 - x_2 + x_4 &= -12 && \text{(II)} \\ x_1 + x_2 + x_5 &= 10 && \text{(III)} \\ x_j &\geq 0, j = 1 \dots 5 \end{aligned}$$

Initiales Tableau:

BV	x_1	x_2	x_3	x_4	x_5	RS
x_3	-1	-1	1			-8
x_4	-3	-1		1		-12
x_5	1	1			1	10
z	-2	-1	0	0	0	0

Die Basislösung $\mathbf{x} = (0, 0, -8, -12, 10)$ verletzt die Nichtnegativitätsrestriktion, sie ist unzulässig.

Primaler Simplex nicht anwendbar, daher *dualer Simplex*.

Tableau vor 1. Iteration:

BV	x_1	x_2	x_3	x_4	x_5	RS
x_3	-1	-1	1	0	0	-8
x_4 x_4	-3	-1 -1	0	1	0	-12
x_5	1	1	0	0	1	10
z	-2	-1	0	0	0	0
$\frac{c_j}{a_{sj}}$	$-\frac{2}{-3}$	$\frac{-1}{-1}$ $-\frac{1}{-1}$	—	—	—	

Tableau nach 1. Iteration:

	BV	x_1	x_2	x_3	x_4	x_5	RS	
(I')	x_3	2	0	1	-1	0	4	(I) + (II')
(II')	x_2	3	1	0	-1	0	12	(II) $\cdot (-1)$
(III')	x_5	-2	0	0	1	1	-2	(III) - (II)
(IV')	z	1	0	0	-1	0	12	(IV) + (II')

Tableau vor 2. Iteration:

BV	x_1	x_2	x_3	x_4	x_5	RS
x_3	2	0	1	-1	0	4
x_2	3	1	0	-1	0	12
x_5 x_5	-2	-2	0	1	1	-2
z	1	0	0	-1	0	12
$\frac{c_j}{a_{sj}}$	$\frac{1}{-2}$	$\frac{1}{-2}$	—	—	—	—

Tableau nach 2. Iteration:

	BV	x_1	x_2	x_3	x_4	x_5	RS	
(I'')	x_3	0	0	1	0	1	2	(I') - 2 (III')
(II'')	x_2	0	1	0	$\frac{1}{2}$	$\frac{3}{2}$	9	(II') - 3 (III'')
(III'')	x_1	1	0	0	$-\frac{1}{2}$	$-\frac{1}{2}$	1	(III') $\cdot (-\frac{1}{2})$
(IV'')	z	0	0	0	$-\frac{1}{2}$	$\frac{1}{2}$	11	(IV') - (II'')

Basislösung (1, 9, 2, 0, 0) ist *zulässig* (nicht zwingend optimal) / die RS enthält keine negativen Einträge, der *dual Simplex bricht ab*.

Tableau 3 ist nun mit dem primalen Simplex zu lösen!

Wiederholung

Primales Simplex-Verfahren

Das Simplex-Tableau

Der primale Simplex-Algorithmus

Der duale Simplex-Algorithmus

102 Vollständiges Beispiel

Zusammenfassung & Ausblick

Schema:

- ▶ Überführe LP in Normalform und stelle Simplex-Tableau auf.
- ▶ Aus Tableau ist Basislösung ablesbar.
- ▶ Falls *Basislösung unzulässig*, verwende **dualen Simplex**.
- ▶ Dualer Simplex berechnet zulässige Basislösung.
- ▶ Falls *Basislösung zulässig*, verwende **primalen Simplex**.

Frage: Woran erkennt man im Tableau eine **zulässige Basislösung**?

- ▶ Alle b_i der RS des Tableaus sind positiv.
- ▶ Negative b_i -Werte bedeuten eine unzulässige Basislösung.

Fragen zur Wiederholung:

- ▶ Wie kann ein LP mit *zulässiger* Basislösung mit dem **primalem Simplex-Verfahren** gelöst werden?
- ▶ Wie kann mit dem **primalem Simplex-Verfahren** für ein LP mit *unzulässiger* Basislösung eine *zulässige* Basislösung berechnet werden?

Klausurrelevante Literatur

Domschke/Drexl: Kapitel 2.2 und 2.3, oder Nickel/Stein/Waldman Kapitel 1.4 und 1.5, oder Werner Kapitel 2.1 und 2.2.2, oder Suhl/Mellouli Kapitel 2.2 und 2.3

Ausblick

- ▶ Tutorium: Primaler/Dualer Simplex
- ▶ Mehrkriterielle Optimierung: Domschke/Drexl Kapitel 2.7, oder Nickel/Stein/Waldman Kapitel 2.3, oder Suhl/Mellouli Kapitel 4.8

Wiederholung

Mehrkriterielle
Entscheidungsprob-
leme

Vektoroptimierung

Präferenzbasierte
LösungsansätzeZusammenfassung &
Ausblick

Teil IV

Lineare Optimierung III – Mehrzieloptimierung

Basis

Jede nichtsinguläre $m \times m$ -Matrix **B** eines LP in Normalform heiSSt **Basis** und besteht aus den linear unabhängigen Spalten von **B**.

Basis-/Nichtbasisvariablen

Die m Spalten von **B** definieren die **Basisvariablen (BV)**.
Die restlichen n Variablen heiSsen **Nichtbasisvariablen (NBV)**.

Basislösung

Setze NBV auf 0. Dann nennen wir die erhaltene eindeutige Lösung für eine Basis **B** **Basislösung**.

Zulässige Basislösung

Eine Basislösung ist **zulässig**, falls alle BV der Basislösung nichtnegative Werte annehmen, sonst unzulässig.

106 Wiederholung

Mehrkriterielle
Entscheidungsprob-
leme

Vektoroptimierung

Präferenzbasierte
Lösungsansätze

Zusammenfassung &
Ausblick

Gegeben: LP in Normalform mit Schlupfvariablen x_3, x_4, x_5 :

$$\begin{aligned} \max z &= 3x_1 + 4x_2 + 0x_3 + 0x_4 + 0x_5 \\ 3x_1 + 2x_2 + x_3 &= 1200 & \text{(I')} \\ 5x_1 + 10x_2 + x_4 &= 3000 & \text{(II')} \\ 0.5x_2 + x_5 &= 125 & \text{(III')} \\ x_1, x_2, x_3, x_4, x_5 &\geq 0 & \text{(IV')} \end{aligned}$$

BV	Strukturvariablen		Schlupfvariablen			RS
	x_1	x_2	x_3	x_4	x_5	
x_3	3	2	1	0	0	1200
x_4	5	10	0	1	0	3000
x_5	0	0.5	0	0	1	125
z	-3	-4	0	0	0	0

Zielfunktion umformen: $z = 3x_1 + 4x_2 \rightarrow z - 3x_1 - 4x_2 = 0$

Basisvariablen: Position der Einheitsvektoren

Achtung: Es gilt $x_i \geq 0$; wird nicht in Tableau vermerkt.

Prinzip: Erzeuge neue Basislösung durch Basistausch

- ▶ Pivotspalte: NBV, die in Basis aufzunehmen ist
- ▶ Pivotzeile: BV, die Basis verlässt
- ▶ Tableau-Update: Mittels GauSS-Jordan in Pivotspalte *Einheitsvektor* konstruieren, das Pivotelement erhält den Wert 1

Primaler Simplex:

Voraussetzung: **Zulässige Basislösung** bekannt

Ablauf: Wähle zuerst Pivotspalte, dann Pivotzeile

Ergebnis: **optimale** Basislösung (sofern eine existiert)

Dualer Simplex:

Voraussetzung: LP in Normalform

Ablauf: Wähle zuerst Pivotzeile, dann Pivotspalte

Ergebnis: zulässige Basislösung (sofern eine existiert)

Mehrkriterielle Entscheidungsprobleme

Vektoroptimierung

Graphische Veranschaulichung

Zielbeziehungen

Pareto-optimale Lösungen

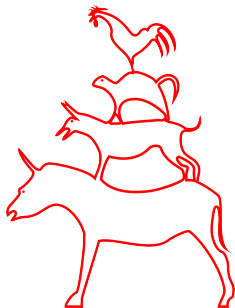
Präferenzbasierte Lösungsansätze

Skalarisierung der Zielfunktion

Lexikographische Zielgewichtung

Zieldominanz

Goal Programming



1. Beispiel: Für welchen Job soll ich mich bewerben?

Problemstellung: Bewerbung für ersten Job nach dem Studium

Mögliche Kriterien:

- ▶ Gehalt
- ▶ Geographische Lage
- ▶ Aufstiegsperspektiven
- ▶ Gestaltungsmöglichkeiten
- ▶ Inhaltliche Zufriedenheit
- ▶ Persönliche Freiheit
- ▶ Verantwortung
- ▶ ...

2. Beispiel: Zusammenstellung Wertpapier-Portfolio

Problemstellung:

Eine **Anlagesumme** soll auf verschiedene **Anlagemöglichkeiten** aufgeteilt werden

Anlagemöglichkeiten: z.B. Aktien, Anleihen, Optionen, Fonds ...

Die **Rendite** einer Anlagemöglichkeit ist **unbekannt, da zufällig**.

hohe Rendite → höheres Risiko;

niedrigere Rendite → geringeres Risiko;

Ein Portfolio-Optimierungsproblem:

max Rendite

min Risiko

u.d.N. Wert der ausgewählter Anlagen = Anlagesumme

Problemstellung: Welches Auto soll man sich kaufen?

Mögliche Kriterien:

- ▶ Preis
- ▶ Design
- ▶ Verbrauch
- ▶ Leistung
- ▶ Sonderausstattung
- ▶ ...

4. Beispiel: Fachliche Ausrichtung eines Krankenhauses

Problemstellung:

Resourcesnutzung eines Krankenhauses soll optimiert werden

Resources: Ärzte verschiedener Fachrichtungen und Qualifikation, Pflegepersonal, Medizinisches Gerät, Krankenzimmer, Operationssäle

Beschränkungen:

- ▶ Neben zwingenden Notfällen auch viele *geplante* Behandlungen
- ▶ Geplante Behandlung erlaubt *fachliche Spezialisierung*

Zielfunktion:

- ▶ min Kosten
- ▶ max Erlöse
- ▶ min Aufenthaltsdauer Patienten im Krankenhaus
- ▶ max Anzahl behandelter Patienten
- ▶ ...

Bei Entscheidungen sind in der Realität häufig **mehrerer Zielkriterien** relevant:

1. Aus einer **gegebenen Menge von Alternativen** ist die **beste** Alternative auszuwählen. Für jede Alternative ist bekannt, inwieweit die Zielkriterien erfüllt werden.
→ *Multi-Attribute Decision Making (MADM)*
2. Restriktionen schränken Werte von Entscheidungsvariablen ein. Die **Werte der Zielkriterien** hängen von den gewählten Werten der Entscheidungsvariablen ab.
→ **Multiobjective (Linear) Optimization (MLP)**

Wir betrachten nur MLP-Probleme; diese sind strukturell schwieriger.

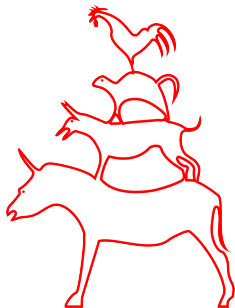
Mehrkriterielle Entscheidungsprobleme

Vektoroptimierung

Graphische Veranschaulichung
Zielbeziehungen
Pareto-optimale Lösungen

Präferenzbasierte Lösungsansätze

Skalarisierung der Zielfunktion
Lexikographische Zielgewichtung
Zieldominanz
Goal Programming



Statt eines (skalaren) Ziels ist ein **Vektor von Zielen** zu optimieren.

$$\begin{array}{rcllcl} \max z_1 = f_1(\mathbf{x}) & = & +3x_1 & + & x_2 & & \\ \max z_2 = f_2(\mathbf{x}) & = & -2x_1 & + & x_2 & & \\ \text{u.d.N.} & & -x_1 & + & x_2 & \leq & 3 \\ & & & + & x_2 & \leq & 4 \\ & & x_1 & + & x_2 & \leq & 6 \\ & & x_1 & & & \leq & 5 \\ & & & & x_j & \geq & 0 \quad j = 1, 2 \end{array}$$

z_i sei der Wert der Zielfunktion $f_i(\mathbf{x})$ mit $i = 1, 2$.

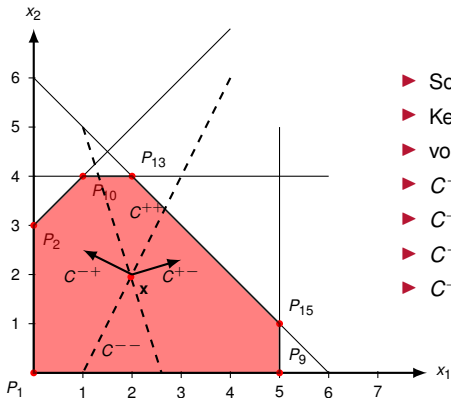
Mehrkriterielles lineares Optimierungsproblem (MLP)

Wir nennen das Problem

$$\max f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_p(\mathbf{x}) \end{pmatrix} \text{ u.d.N. } \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$$

ein mehrkriterielles lineares Optimierungsproblem, also ein LP mit vektorwertiger Zielfunktion.

Isogewinngeraden und Gradienten für f_1 und f_2



- ▶ Schnittpunkt Iso-Geraden \mathbf{x}
- ▶ Kegel C^{--} , C^{-+} , C^{+-} , C^{++}
- ▶ von \mathbf{x} in Richtung ...
- ▶ C^{--} : z_1 **und** z_2 **schlechter**
- ▶ C^{++} : z_1 **und** z_2 **besser**
- ▶ C^{-+} : z_1 **schlechter**, z_2 **besser**
- ▶ C^{+-} : z_1 **besser**, z_2 **schlechter**

Die Menge zulässiger Lösungen eines MLP sei M .

Sei z_i der Wert der Zielfunktion $f_i(\mathbf{x})$, $\mathbf{x} \in M$.

Sei z_i^{opt} der optimale Wert des Ziels i , d.h. $z_i^{opt} = f_i(\mathbf{x}^*)$, $\mathbf{x}^* \in M$.

Zielerreichungsgrad

Der **Zielerreichungsgrad** Z_i des Ziels i ist gegeben durch

$$Z_i = \frac{z_i}{z_i^{opt}}$$

Beispiel Krankenhaus

Optimale Nutzung des OP (unter Restriktionen): **327 Tage**

Erreichte Nutzung des OP: **312 Tage**

Zielerreichungsgrad: $\frac{312}{327} = \mathbf{95.4\%}$

Operations Research

J. Pannek

Wiederholung

Mehrkriterielle
Entscheidungsprobleme

Vektoroptimierung

Graphische
Veranschaulichung

116 Zielbeziehungen

Pareto-optimale Lösungen

Präferenzbasierte
Lösungsansätze

Zusammenfassung &
Ausblick

Zwei Ziele sind

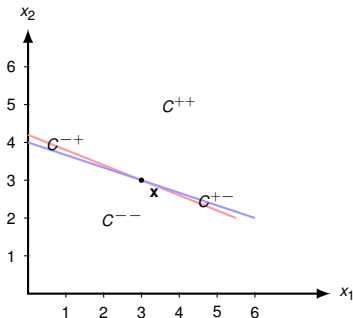
... **komplementär**, wenn ein **höherer** Zielerreichungsgrad des einen Ziels zu einem **höheren** Zielerreichungsgrad des anderen Ziels führt.

... zueinander **neutral**, wenn sie sich in ihrem jeweiligen Zielerreichungsgrad **gegenseitig nicht beeinflussen**.

... **konkurrierend**, wenn ein **höherer** Zielerreichungsgrad des einen Ziels zu einem **geringeren** Zielerreichungsgrad des anderen Ziels führt.

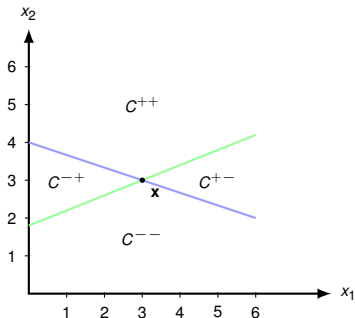
Beziehungen zwischen Zielen 3/3 – Stärke Zielkonflikt

$$\begin{aligned} \max z_1 &= 2x_1 + 5x_2 \\ \max z_2 &= 2x_1 + 6x_2 \end{aligned}$$



C^{++} Kegel weit
geringer Zielkonflikt

$$\begin{aligned} \max z_1 &= -2x_1 + 5x_2 \\ \max z_2 &= 2x_1 + 6x_2 \end{aligned}$$



C^{++} Kegel schmal
starker Zielkonflikt

Pareto-optimale Lösung

Ein Bewegung in Richtung des verbessernden Kegels ist nicht möglich, ohne die Zulässigkeit einzubüßen.

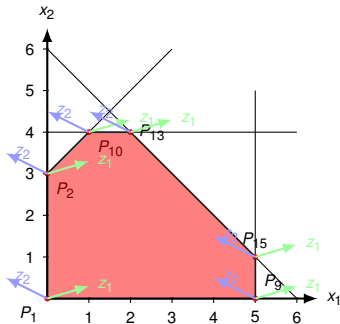
Pareto Front

Die Menge aller Pareto-optimalen Zielfunktionsvektoren heißt **Pareto-Front**.

Ziel des Vektoroptimierungs-Problems

Ziel eines Vektoroptimierungs-Problems ist die **Berechnung der Pareto-Front**.

$\max z_1 = f_1(\mathbf{x})$	$=$	$+3x_1$	$+$	x_2	
$\max z_2 = f_2(\mathbf{x})$	$=$	$-2x_1$	$+$	x_2	
u.d.N.		$-x_1$	$+$	x_2	≤ 3
			$+$	x_2	≤ 4
		x_1	$+$	x_2	≤ 6
		x_1			≤ 5
		x_1		x_2	≥ 0



P_1, P_9 :
Bessere und zulässige Lösungen
existieren

$P_2, P_{10}, P_{13}, P_{15}$:
Bessere Lösungen sind *stets*
unzulässig

Pareto Front: Gebildet durch Punkte
der Grenze $P_2, P_{10}, P_{13}, P_{15}$

Mehrkriterielle Entscheidungsprobleme

Vektoroptimierung

Graphische Veranschaulichung

Zielbeziehungen

Pareto-optimale Lösungen

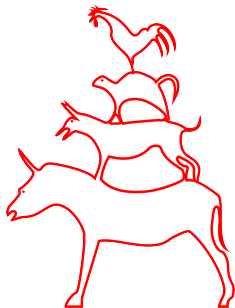
Präferenzbasierte Lösungsansätze

Skalarisierung der Zielfunktion

Lexikographische Zielgewichtung

Zieldominanz

Goal Programming



Präferenz

Vorliebe und Abneigung für Zielkriterien werden Präferenzen genannt.

Mittels **Präferenzen des Entscheidungsträgers** wird das Mehrziel– in ein Einziel–Optimierungsproblem überführt.

vektorwertige Zielfunktion \implies **skalarwertige** Zielfunktion

Vorteile:

- ▶ Änderungen nur auf **Modellebene**
- ▶ Simplex-Verfahren anwendbar

Nachteile:

- ▶ Präferenzen müssen **bekannt** sein
- ▶ Präferenzen sind stets **subjektiv**
- ▶ Nur **subjektiv optimale** Lösung, keine Pareto-Front, keine Trade-Offs zwischen Lösungen

Operations Research

J. Pannek

Wiederholung

Mehrkriterielle
Entscheidungsprob-
leme

Vektoroptimierung

121 **Präferenzbasierte
Lösungsansätze**

Skalarisierung der
Zielfunktion
Lexikographische
Zielgewichtung
Zieldominanz
Goal Programming

Zusammenfassung &
Ausblick

Diagnosis-Related-Group (DRG)

Krankenkasse zahlt DRG-abhängigen Festpreis pro Patient

Wirtschaftliches Ergebnis hängt von Behandlungskosten ab

Bezeichne Anzahl Patienten in DRG i , $i = 1, 2, 3$ mit x_i .

Lineare Restriktionen:

$$10x_1 + 4x_2 + 4x_3 \leq 3000 \quad \text{Pflegetage}$$

$$5x_1 + 4x_2 + 6x_3 \leq 1800 \quad \text{OP-Stunden}$$

$$3.000x_1 + 2.000x_2 + 2.500x_3 \leq 904.000 \quad \text{Budget}$$

$$x_i \geq 0 \quad i = 1, 2, 3$$

Zielfunktionen:

$z_1 = f_1(\mathbf{x})$: Max. Deckungsbeitrag

$z_2 = f_2(\mathbf{x})$: Max. Anzahl behandelter Patienten

$z_3 = f_3(\mathbf{x})$: Max. Anzahl behandelter Patienten in DRG-Fallgruppe 1

$z_4 = f_4(\mathbf{x})$: Max. die Auslastung der Operationsäle

$$\max f_1(\mathbf{x}) = 1.200x_1 + 800x_2 + 1100x_3$$

$$\max f_2(\mathbf{x}) = x_1 + x_2 + x_3$$

$$\max f_3(\mathbf{x}) = x_1$$

$$\max f_4(\mathbf{x}) = 5x_1 + 4x_2 + 6x_3$$

u.d.N. Einhaltung der linearen Restriktionen, d.h.
Pflegetage, OP-Stunden, Budget, Nichtnegativität.

Skalarisierung

Für jede der p Zielfunktionen $f_i(\mathbf{x})$ wird ein (subjektiver) Gewichtungsfaktor $\lambda_i \geq 0$ bestimmt.

$$\text{Ersetze } f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_p(\mathbf{x}) \end{pmatrix} \text{ durch } f^{gew}(\mathbf{x}) = \lambda_1 \cdot f_1(\mathbf{x}) + \dots + \lambda_p \cdot f_p(\mathbf{x})$$

$\implies f^{gew}(\mathbf{x})$ ist *skalarwertig*, die Restriktionen unverändert

Vorteil: leicht verständlich

Nachteil: Normierung der Ziele schwierig (etwa bei Kosten und Kundenzufriedenheit)

Zielgewichte festlegen:

$$\lambda_1 = 1; \lambda_2 = 4; \lambda_3 = 0.5; \lambda_4 = 2$$

⇒ Ziel 2 ist viermal so wichtig wie Ziel 1 und achtmal so wichtig wie Ziel 3.

Neues Einziel-Optimierungsmodell:

$$\begin{array}{ll} \max & f^{gew} = 1 \cdot f_1(\mathbf{x}) + 4 \cdot f_2(\mathbf{x}) + 0.5 \cdot f_3(\mathbf{x}) + 2 \cdot f_4(\mathbf{x}) \\ \text{u.d.N.} & \text{Einhaltung der Restriktionen} \end{array}$$

Zur Erinnerung:

f_1 Deckungsbeitrag, f_2 Anz. Patienten, f_3 Anz. Patienten DRG 1, f_4 Auslastung OP

Beispiel Medallienspiegel

Goldmedaillen **unendlich** wichtiger als Silbermedaillen

Aktueller Medallienspiegel

R.	Land	Abk.	G.	S.	B.
1.	Vereinigte Staaten	USA	46	29	29
2.	Volksrepublik China	CHN	38	27	23
3.	Vereinigtes Königreich	GBR	29	17	19
4.	Russland	RUS	24	26	32
5.	Südkorea	KOR	13	8	7
6.	Deutschland	GER	11	19	14
7.	Frankreich	FRA	11	11	12
8.	Italien	ITA	8	9	11
9.	Ungarn	HUN	8	4	5
10.	Australien	AUS	7	16	12
11.	Japan	JPN	7	14	17
12.	Kasachstan	KAZ	7	1	5

Beispiele Lexikon

Erster Buchstabe **unendlich** wichtiger als folgende

Operations Research

J. Pannek

Wiederholung

Mehrkriterielle
Entscheidungsprob-
leme

Vektoroptimierung

Präferenzbasierte
Lösungsansätze

Skalarisierung der
Zielfunktion

126 Lexikographische
Zielgewichtung

Zieldominanz
Goal Programming

Zusammenfassung &
Ausblick

Lexikographischer Zielgewichtung

Ziele werden nach **Wichtigkeit geordnet** und sukzessive als **Nebenbedingungen** eingefügt.

- 1. Schritt:** Ziele werden nach Wichtigkeit geordnet
- 2. Schritt:** Löse LP
- 3. Schritt:** Existiert nur eine optimale Lösung, beende Verfahren.
- 4. Schritt:** Füge neue Restriktion $f_i(\mathbf{x}) \geq \bar{z}_i$ ein, ersetze Ziel durch das nächstwichtigste und gehe zu Schritt 2.

Vorteil: leicht verständlich

Nachteil: Verschiedene Ergebnisse je Ordnung

Start:

Rangfolge der Ziele festlegen, z.B. f_3, f_1, f_2, f_4

1. Iteration:

$$\bar{z}_3 = f_3(\mathbf{x}) = \max f_3(\mathbf{x}) \text{ u.d.N. Einhaltung Restriktionen}$$

2. Iteration:

$$\bar{z}_1 = \max f_1(\mathbf{x}) \text{ u.d.N. Restriktionen und } f_3(\mathbf{x}) \geq \bar{z}_3$$

3. Iteration:

$$\bar{z}_2 = \max f_2(\mathbf{x}) \text{ u.d.N. Restriktionen und } f_1(\mathbf{x}) \geq \bar{z}_1; f_3(\mathbf{x}) \geq \bar{z}_3$$

usw. bis das zu lösende LP **keine oder nur eine zulässige** Lösung hat.

Zur Erinnerung:

f_1 Deckungsbeitrag, f_2 Anz. Patienten, f_3 Anz. Patienten DRG 1, f_4 Auslastung OP

Zieldominanz

Bestimme aus den p Zielen ein **alleiniges Hauptziel** $f_h(\mathbf{x})$.
Übrige Ziele werden zu **Nebenzielen**, die in Form von Restriktionen ein unteres und/oder oberes **Anspruchsniveau** erfüllen müssen.

$$\begin{array}{ll} \max & \text{Hauptzielfunktion } f_h(\mathbf{x}) \\ \text{u.d.N.} & \text{allgemeine Restriktionen} \\ & f_i(\mathbf{x}) \geq \text{unteres Anspruchsniveau} \quad \exists i \text{ mit } f_i(\mathbf{x}) \neq f_h(\mathbf{x}) \\ & f_i(\mathbf{x}) \leq \text{oberes Anspruchsniveau} \quad \exists i \text{ mit } f_i(\mathbf{x}) \neq f_h(\mathbf{x}) \end{array}$$

Vorteil: leicht verständlich

Nachteil: Zu ehrgeizige Anspruchsniveaus können die Menge der zulässigen Lösungen zu stark einschränken (keine zulässige Lösung).

Schritt 1: Lege Hauptziel fest, z.B. $f_2(\mathbf{x})$

Schritt 2: Definiere obere/untere Anspruchsniveaus, z.B.

Deckungsbeitrag mindestens $\underline{z}_1 = 300.000$ GE

Anzahl behandelter Patienten DRG 1 zwischen $\underline{z}_3 = 100$ und $\bar{z}_3 = 150$

Mindestauslastung OP $\underline{z}_4 = 80\%$

Neues Optimierungsproblem:

$$\begin{array}{ll} \max & f_2(\mathbf{x}) \\ \text{u.d.N.} & \text{Einhaltung der Restriktionen} \\ & f_1(\mathbf{x}) \geq 300.000 \\ & f_3(\mathbf{x}) \geq 100 \\ & f_3(\mathbf{x}) \leq 150 \\ & f_4(\mathbf{x}) \geq 0.80 \end{array}$$

Zur Erinnerung:

f_1 Deckungsbeitrag, f_2 Anz. Patienten, f_3 Anz. Patienten DRG 1, f_4 Auslastung OP

Goal Programming

Goal Programming (GP) kombiniert die Ideen der

- ▶ Zielgewichtung und der
- ▶ Zieldominanz.

Die Modellierungstechnik unterstützt

- ▶ klassischen max/min Zielen, aber auch
- ▶ das **möglichst präzise Erreichen** eines gegebenen Zielwerts sowie
- ▶ **weiche Restriktionen** (im Unterschied zu klassischen, harten) Restriktionen.

Vokabeln:

objective (function)	Zielfunktion
target value	Zielwert, Anspruchsniveau
goal	Zielmarke, Zweck - eher nicht i.S. einer Zielfunktion

Wiederholung

Mehrkriterielle
Entscheidungspro-
bleme

Vektoroptimierung

Präferenzbasierte
Lösungsansätze

Skalarisierung der
Zielfunktion

Lexikographische
Zielgewichtung

Zieldominanz

131 Goal Programming

Zusammenfassung &
Ausblick

1. Schritt: Definiere für **jedes Ziel** f_i ein Anspruchsniveau (goal) g_i .

2. Schritt: Führe **zwei Abweichungsvariablen** (deviation) für jedes Ziel f_i ein:

- ▶ d_i^- misst **Unterschreiten** des Anspruchsniveau
- ▶ d_i^+ misst **Überschreiten** des Anspruchsniveaus

Mit d_i^- und d_i^+ wird jedes Ziel als **Gleichungs-Restriktion** aufgefasst.

Neue Zielfunktion:

$$\min \sum \text{unerwünschten Abweichungen vom Anspruchsniveau}$$

Definiere Anspruchsniveaus für alle Ziele:

Deckungsbeitrag mindestens $g_1 = 300.000$ GE

Anzahl Patienten $g_2 = 700$

Anzahl Patienten in DRG 1 $g_3 = 125$

Auslastung OP $g_4 = 80\%$

Goal-Programming Problem:

min unerwünschte Abweichungen von Goals

u.d.N. Einhaltung der Restriktionen

$f_1(\mathbf{x}) + d_1^- - d_1^+ =$	300.000	Deckungsbeitrag
$f_2(\mathbf{x}) + d_2^- - d_2^+ =$	700	Behandelte Patienten
$f_3(\mathbf{x}) + d_3^- - d_3^+ =$	125	Behandelte Patienten in DRG 1
$f_4(\mathbf{x}) + d_4^- - d_4^+ =$	0.80	Auslastung OP

min	unerwünschte Abweichungen von Goals		
u.d.N.	Einhaltung der Restriktionen		
$f_1(\mathbf{x}) + d_1^- - d_1^+ =$	300.000	Deckungsbeitrag	
$f_2(\mathbf{x}) + d_2^- - d_2^+ =$	700	Behandelte Patienten	
$f_3(\mathbf{x}) + d_3^- - d_3^+ =$	125	Behandelte Patienten in DRG 1	
$f_4(\mathbf{x}) + d_4^- - d_4^+ =$	0.80	Auslastung OP	

Frage: Wie ist die Zielfunktion zu modellieren?

a) Alle vier Ziele sind zu maximieren:

$$\min d_1^- + d_2^- + d_3^- + d_4^-$$

b) Maximiere f_1 und f_3 . Die Goals für f_2 und f_4 sind möglichst zu erreichen:

$$\min d_1^- + (d_2^- + d_2^+) + d_3^- + (d_4^- + d_4^+)$$

c) Alle Goal-Abweichungen sind zu minimieren:

$$\min (d_1^- + d_1^+) + (d_2^- + d_2^+) + (d_3^- + d_3^+) + (d_4^- + d_4^+)$$

Beispiel: Mit *Normierung* und *Gewichtung*

$$\min \lambda_1 \cdot d_1^- + \lambda_2 \cdot (d_2^- + d_2^+) + \lambda_3 \cdot d_3^- + \lambda_4 \cdot (d_4^- + d_4^+)$$

Restriktionen

Eine **harte Restriktion** ist in jedem Fall **einzuhalten** (klassischer Fall).

Eine **weiche Restriktion** darf verletzt werden. Die **Verletzung** soll **so gering wie möglich** sein.

In der Realität sind viele Restriktionen weich, z.B.

- ▶ Anzahl Mitarbeiter – Überstunden, Werkverträge, Zeitarbeit
- ▶ Budget – Kunst des Rechnungswesen

Modellierung weicher Restriktionen:

Eine harte Restriktion ist mit Hilfe von Abweichungsvariablen $d^- \geq 0$ und $d^+ \geq 0$ wie folgt umzuformen:

Situation	Formulierung Goal-Restriktion	Beitrag zur Zielfunktion
$LS \leq RS$	$LS + d^- - d^+ = RS$ $d^-, d^+ \geq 0$	$\min d^+$
$LS = RS$		$\min d^- + d^+$
$LS \geq RS$		$\min d^-$

LS: linke Seite des Restriktionssystems; **RS:** rechte konstante/Goal Seite.

Unterschreiten des Anspruchsniveaus $\rightarrow d_i^-$

Überschreiten des Anspruchsniveaus $\rightarrow d_i^+$

Beispiel Budget-Restriktion

100 Euro können für Lebensmittel (x_L) oder für Unterhaltung (x_U) ausgegeben werden.

$$x_L + x_U \leq 100 \quad \text{harte Restriktion}$$

$$x_L + x_U + d^- - d^+ = 100 \quad \text{weiche / Goal Programming Restriktion}$$

Unterschreiten des Budgets: $x_L = 60, x_U = 30 \rightarrow d^- = 10, d^+ = 0$

Überschreiten des Budgets: $x_L = 70, x_U = 50 \rightarrow d^+ = 20, d^- = 0$

Budget-Überschreitungen sollen **so gering wie möglich** ausfallen, daher **Änderung Zielfunktion**.

Größere Werte für d^+ sind schlechter. In geeigneter Weise berücksichtigen, z.B. $\min d^+$.

Fragen zur Wiederholung:

- ▶ Was bedeutet der Begriff Vektoroptimierung?
- ▶ Wie sind die Begriffe Zielbeziehung, Pareto-Front, lexikographische Zielgewichtung, Skalarisierung, Zieldominanz und Goal-Programming einzuordnen und anzuwenden?

Klausurrelevante Literatur

Domschke/Drexl: Kapitel 2.7, oder Nickel/Stein/Waldman Kapitel 2.3

Ausblick

- ▶ Tutorium: Vektoroptimierung, Zieldominanz, Pareto Front
- ▶ Einführung Graphentheorie: Domschke/Drexl Kapitel 3.1 und 3.2, oder Nickel/Stein/Waldman Kapitel 3.1 und 3.2, oder Werners Kapitel 5

Wiederholung

Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

Berechnung kürzester
Wege in Graphen

Anwendungsmöglichkeiten
kürzester Wege

Zusammenfassung &
Ausblick

Teil V

Graphentheorie I – Kürzeste Wege in Graphen

Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

Berechnung kürzester
Wege in Graphen

Anwendungsmöglichkeiten
kürzester Wege

Zusammenfassung &
Ausblick

Modelle für einige ausgewählte Problemtypen:

- ▶ Mischungsprobleme
- ▶ Produktionsprogrammplanung

Regeln zur Modelltransformation:

- ▶ Standardform
- ▶ Normalform

Mehrere Zielkriterien:

- ▶ Zielgewichtung (skalarwertig, lexikographisch)
- ▶ Zieldominanz
- ▶ Goal-Programming

Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

Berechnung kürzester
Wege in Graphen

Anwendungsmöglichkeiten
kürzester Wege

Zusammenfassung &
Ausblick

Grafisches Lösungsverfahren für LP:

- ▶ Einziel LP
- ▶ Mehrziel LP

Simplex-Algorithmus für beliebige LP

- ▶ Transformation
- ▶ Primaler Simplex
- ▶ Dualer Simplex

141 **Wiederholung**

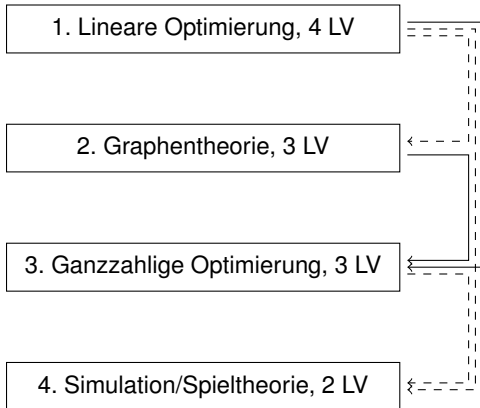
Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

Berechnung kürzester
Wege in Graphen

Anwendungsmöglichkeiten
kürzester Wege

Zusammenfassung &
Ausblick



Beispiele für Netzwerkmodelle basierend auf Graphen

Modellierung mittels Graphen

Grundlegende Strukturen

Repräsentation von Graphen mit Matrizen

Berechnung kürzester Wege in Graphen

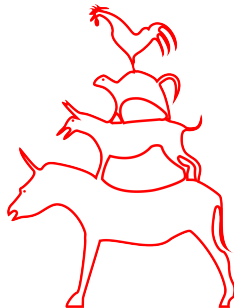
Problem

Funktionsprinzip

Lösung mittels Dijkstra-Algorithmus

Lösung mittels Simplex-Algorithmus

Anwendungsmöglichkeiten kürzester Wege

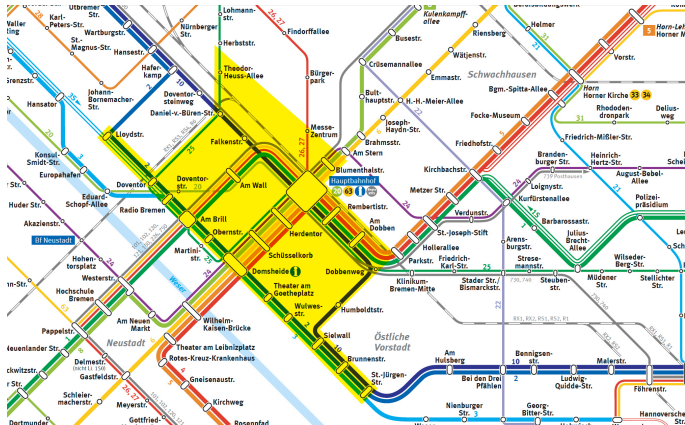




Telefonnetz, Internet, Strom-/Gasnetz, Wasserversorgungssystem

Modelle physischer Strukturen

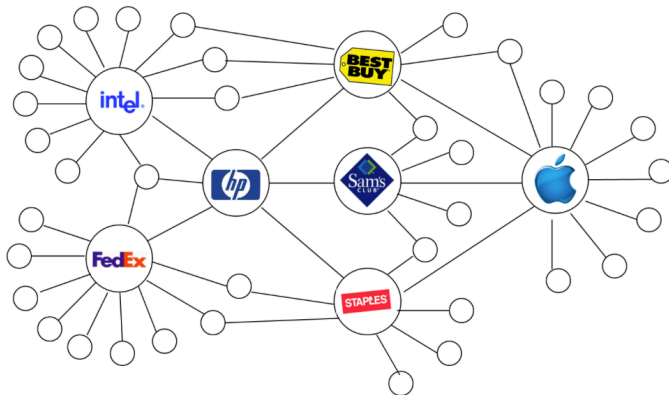
Netz von Haltestellen, Bus und Straßenbahn Bremen



Bestandteile:

Knoten: Haltestellen

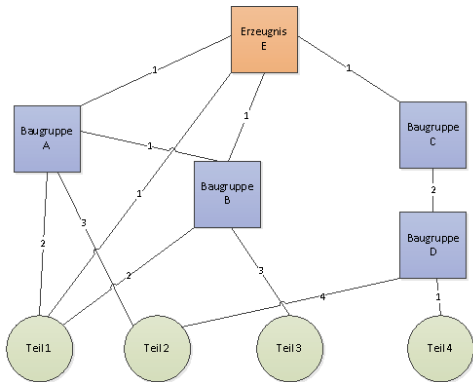
Kanten: Verbindungsstrecken



Bestandteile:

Knoten: Lieferanten

Kanten: Beziehungen/Verträge



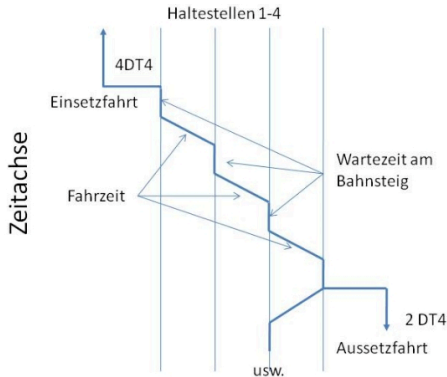
Bestandteile:

Knoten: Erzeugnisse, Baugruppen, und Einzelteile

Kanten: Konstruktionszugehörigkeiten

Modelle räumlich-zeitlicher Abhängigkeiten

Umlaufplanung im ÖPNV



Bestandteile:

Knoten: Ankunfts- oder Abfahrtsereignis

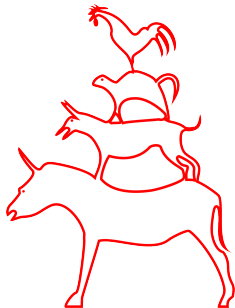
Kanten: Verbindungsweg/-zeit

Beispiele für Netzwerkmodelle basierend auf Graphen

Modellierung mittels Graphen Grundlegende Strukturen Repräsentation von Graphen mit Matrizen

Berechnung kürzester Wege in Graphen Problem Funktionsprinzip Lösung mittels Dijkstra-Algorithmus Lösung mittels Simplex-Algorithmus

Anwendungsmöglichkeiten kürzester Wege



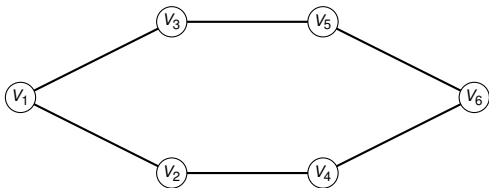
Graphen repräsentieren **Beziehungen** zwischen **Objekten**.

Definition Graph

- ▶ Ein **Graph** G ist ein Paar von zwei Mengen $G = (V, E)$.
- ▶ $V = \{v_1, \dots, v_n\}$ ist eine **nichtleere Menge** deren Elemente man als **Knoten** bezeichnet.
- ▶ $E = \{e_1, \dots, e_m\}$ ist eine Menge deren Elemente man als **Kanten** bezeichnet.
- ▶ Jeder **Kante** sind zwei Knoten zugeordnet (Inzidenzabbildung $\omega : E \rightarrow V \times V$).

Grundbegriffe → z.B. Domschke/Drexl, Kapitel 3.1

Beispiel ungerichteter Graph:



Knotenmenge

$$V_1 = \{1, 2, 3, 4, 5, 6\}$$

Kantenmenge

$$E_1 = \{e_1, e_2, e_3, e_4, e_5, e_6\}$$

Inzidenzabbildung

$$\begin{aligned} \omega: e_1 &\rightarrow \{1, 2\}, e_2 \rightarrow \{1, 3\}, \\ e_3 &\rightarrow \{2, 4\}, e_4 \rightarrow \{3, 5\}, \\ e_5 &\rightarrow \{4, 6\}, e_6 \rightarrow \{5, 6\} \end{aligned}$$

Wiederholung

Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

148 Grundlegende Strukturen

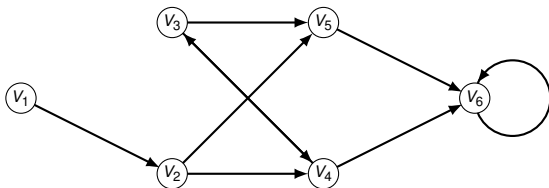
Repräsentation von
Graphen mit Matrizen

Berechnung kürzester
Wege in Graphen

Anwendungsmöglichkeit
kürzester Wege

Zusammenfassung &
Ausblick

Beispiel gerichteter Graph:



Knotenmenge $V_2 = \{1, 2, 3, 4, 5, 6\}$

Kantenmenge $E_2 = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$

Inzidenzabbildung $\omega: e_1 \rightarrow (1, 2), e_2 \rightarrow (2, 4),$
 $e_3 \rightarrow (2, 5), e_4 \rightarrow (3, 4),$
 $e_5 \rightarrow (3, 5), e_6 \rightarrow (4, 3),$
 $e_7 \rightarrow (4, 6), e_8 \rightarrow (5, 6)$
 $e_9 \rightarrow (6, 6)$

Wiederholung

Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

149 Grundlegende Strukturen

Repräsentation von
Graphen mit Matrizen

Berechnung kürzester
Wege in Graphen

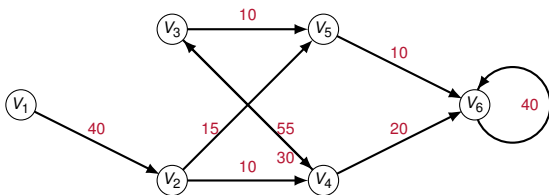
Anwendungsmöglichkeiten
kürzester Wege

Zusammenfassung &
Ausblick

Definition Bewerteter Graph

Ein bewerteter Graph ist ein Graph mit einer **Bewertungsfunktion** $c : V \rightarrow \mathbb{R}$, die jedem Knoten bzw. jeder Kante eine reelle Zahl zuweist.

Beispiel bewerteter Graph:



Notation:

Kosten der Kante (i, j) bezeichnet man mit c_{ij} .

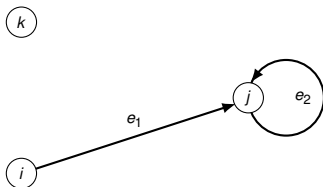
Wiederholung

Beispiele für
Netzwerkmodelle
basierend auf
GraphenModellierung mittels
Graphen

150 Grundlegende Strukturen

Repräsentation von
Graphen mit MatrizenBerechnung kürzester
Wege in GraphenAnwendungsmöglichkeit
kürzester WegeZusammenfassung &
Ausblick

Beispiel bewerteter Graph:



Bezeichnungen:

- ▶ Kante e_1 ist mit Knoten i und j **inzident**
- ▶ Knoten i heiSSt **Nachbar** von j , j heiSSt **Nachbar** von i
- ▶ Knoten i und j sind **benachbart** oder **adjazent**
- ▶ Zahl der mit i inzidenten Kanten heiSSt **Grad** $g(i)$
- ▶ $N(i)$ heiSSt **Menge aller Nachbarn** von Knoten i
- ▶ Knoten k ist **isoliert**
- ▶ Kante e_2 heiSSt **Schlinge**

Vorgänger

In einem gerichteten Graph heißt der Ursprung einer Kante **Vorgänger** und der Abschluss der Kante **Nachfolger**. Es bezeichne $P(i)$ und $S(i)$ die Menge der Vorgänger und Nachfolger von Knoten i .

Parallel

Zwei ungerichtete Kanten sind **parallel**, wenn sie dasselbe Knotenpaar verbinden.

Schlichter Graph

Ein Graph, der **weder Schlingen noch parallele** Kanten bzw. Pfeile besitzt, wird als **schlicht** bezeichnet.

Digraph

Ein schlichter gerichteter Graph mit **endlicher Knotenmenge** heißt Digraph.

Weg

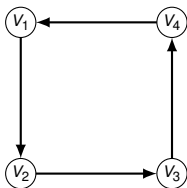
Ein **Weg** bezeichnet eine **Folge von Knoten**, in der **zwei aufeinanderfolgende** Knoten durch eine Kante verbunden sind.

Ein Weg in einem gerichteten Graphen erfordert, dass aufeinanderfolgende Knoten durch **gerichtete** Kanten verbunden sind.

Kreis

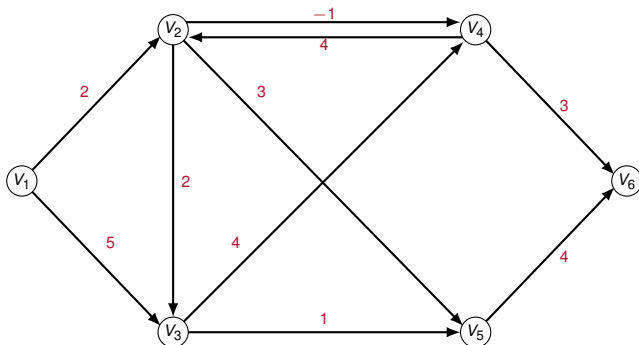
Wenn erster und letzter Knoten eines Wegs übereinstimmen, nennen wir den Weg einen **Kreis**.

Einen Graph ohne Kreis nennt man **kreisfrei**.



Aufgabe (Wege in einem Graph)

Gegeben sei der folgende bewertete Graph:



1. Geben Sie drei verschiedene **Wege** von Knoten V_1 zu Knoten V_6 an.
2. Geben Sie einen **Kreis** beginnend mit Knoten V_4 an.
3. Welches ist der **kostengünstigste** Weg von Knoten V_1 zu Knoten V_6 ?
4. Welches ist der **kostengünstigste** Weg von Knoten V_1 zu Knoten V_6 , falls sich die Kosten c_{42} auf -5 ändern?

Adjazenzmatrix (un)gerichteter Graph

Sei $G = (V, E)$ ein (un)gerichteter, endlicher Graph ohne parallele Kanten. Die zugehörige Adjazenzmatrix $A = (a_{ij})$ ist definiert mit

$$a_{ij} = \begin{cases} 1 & \text{falls } \{i, j\} \in E \\ 0 & \text{sonst.} \end{cases} \quad \text{bzw.} \quad a_{ij} = \begin{cases} 1 & \text{falls } (i, j) \in E \\ 0 & \text{sonst.} \end{cases}$$

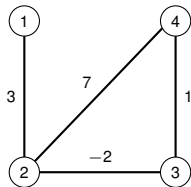
Merke: Adjazenzmatrizen ungerichteter Graphen sind **symmetrisch**.

Adjazenzmatrix mit Kantengewichten

Sei $G = (V, E)$ ein gerichteter, endlicher Graph ohne parallele Kanten mit Kantengewichten. Die zugehörige Adjazenzmatrix $A = (a_{ij})$ ist definiert mit

$$a_{ij} = \begin{cases} c_{ij} & \text{falls } \{i, j\} \in E \text{ (ungerichtet) bzw. } (i, j) \in E \text{ (gerichtet)} \\ \infty & \text{sonst.} \end{cases}$$

Der **bewertete** Graph G als Adjazenzmatrix A :

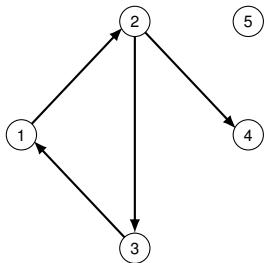


$$A = \begin{pmatrix} 0 & 3 & 0 & 0 \\ 3 & 0 & -2 & 7 \\ 0 & -2 & 0 & 1 \\ 0 & 7 & 1 & 0 \end{pmatrix}$$

G und A sind **strukturgleich**, d.h. Eigenschaften von G sind in A enthalten (und umgekehrt).

Darstellung eines Graph als Matrix 3/3 – Beispiel 2

Ein **gerichteter** Graph G als Adjazenzmatrix A :



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Nachfolger, Vorgänger, Eingangs- und Ausgangsgrad von k

$$\text{Anz. Nachfolger: } |S(k)| = \sum_{j=1}^n a_{kj} (= g^+(k))$$

$$\text{Anz. Vorgänger: } |P(k)| = \sum_{i=1}^n a_{ik} (= g^-(k))$$

Operations Research

J. Pannek

Wiederholung

Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

Grundlegende Strukturen

157

Repräsentation von
Graphen mit Matrizen

Berechnung kürzester
Wege in Graphen

Anwendungsmöglichkeiten
kürzester Wege

Zusammenfassung &
Ausblick

Beispiele für Netzwerkmodelle basierend auf Graphen

Modellierung mittels Graphen

Grundlegende Strukturen

Repräsentation von Graphen mit Matrizen

Berechnung kürzester Wege in Graphen

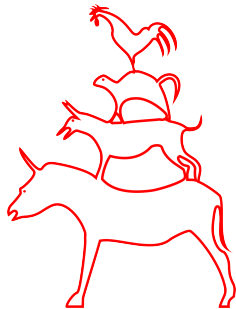
Problem

Funktionsprinzip

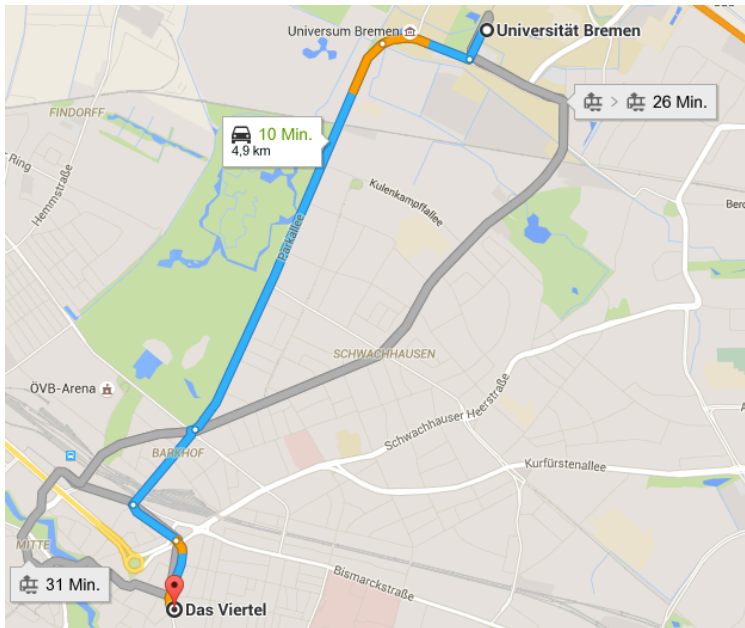
Lösung mittels Dijkstra-Algorithmus

Lösung mittels Simplex-Algorithmus

Anwendungsmöglichkeiten kürzester Wege



Wie funktioniert ein Navigationsdienst?



Operations Research

J. Pannek

Wiederholung

Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

Berechnung kürzester
Wege in Graphen

158 **Problem**

Funktionsprinzip
Lösung mittels
Dijkstra-Algorithmus
Lösung mittels
Simplex-Algorithmus

Anwendungsmöglichkeit
kürzester Wege

Zusammenfassung &
Ausblick

Problemstellung

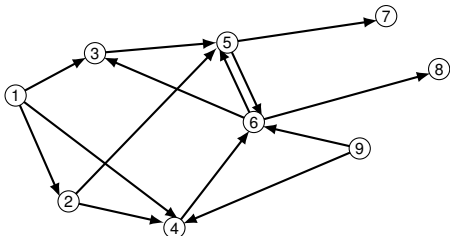
Gesucht ist in einem Graphen *ein kürzester Weg* . . .

1. von **einem Startknoten** s zu **einem Endknoten** t
(single-pair shortest path problem)
2. **einem Startknoten** s zu **allen anderen Knoten** eines Graphen
(single-source shortest path problem)
3. von **allen Knoten** eines Graphen zu **einem Endknoten** t
(single-destination shortest path problem)
4. jeweils zwischen **allen Knotenpaaren**
(all-pairs shortest path problem).

Der **Dijkstra-Algorithmus** löst das single-source shortest path problem.

Minimalität

Ein minimaler Weg besteht aus minimalen Teilwegen.



Annahme: minimaler 1-5-Weg ist 1, 2, 4, 6, 5

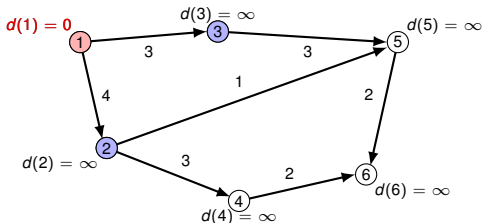
Folgerung: minimaler 1-6-Weg ist 1, 2, 4, 6

Dreiecksungleichung

Falls d_{uv} die kürzeste Distanz zwischen Knoten u und v ist, dann gilt für einen beliebigen Knoten x

$$d_{uv} \leq d_{ux} + d_{xv}.$$

Gegeben: Gerichteter Graph $G = (V, E, c)$ mit **nichtnegativen** Kantengewichten und einem Startknoten, z.B. $1 \in V$.



Anmerkung Handout-Version: die relevanten Details sind nur im animierten Foliensatz erkennbar.

- ▶ Graph $G = (V, E, c)$ mit **nichtnegativen** Kantengewichten
- ▶ Ein **Startknoten** $s \in V$
- ▶ Die **Anzahl der Knoten** im Graph sei $n = |V|$
- ▶ $S(i) = \{j | (i, j) \in E\}$ ist die **Menge der Nachfolger** von Knoten i
- ▶ d ist ein n -Vektor von Distanzen. Die Distanz $d(i)$ ist die bislang bekannte **kürzeste Distanz** von Startknoten s zu Knoten i .
- ▶ R ist ein n -Vektor für Knoten. Der Knoten $R(i)$ ist der **direkter Vorgänger von Knoten** i auf dem kürzesten Weg von s zu i .
- ▶ \mathcal{M} ist eine **Menge von Knoten**. Für jeden Knoten i in \mathcal{M} ist bereits ein Weg (nicht zwingend der kürzeste) von s nach i bekannt. (\mathcal{M} enthält die **blauen Knoten**).

Algorithmus 3: Dijkstra-Algorithmus

Input : Digraph $G = (V, E, c)$ mit $|V| = n$ und $c_{ij} \geq 0$ für alle $(i, j) \in E$,
Startknoten $s \in V$

```
1 begin
2    $\mathcal{M} := \{a\}, d(a) := 0, d(i) := \infty$  für alle  $i \neq a$ 
3   while  $\mathcal{M} \neq \emptyset$  do
4     Bestimme  $h = \underset{i \in \mathcal{M}}{\operatorname{arg\,min}} d(i)$ 
5      $\mathcal{M} := \mathcal{M} \setminus \{h\}$ 
6     foreach  $j \in S(h)$  do
7       if  $d(j) > d(h) + c_{hj}$  then
8         |  $d(j) := d(h) + c_{hj}, R(j) := h, \mathcal{M} := \mathcal{M} \cup \{j\}$ 
9       end
10    end
11  end
12 end
```

Output: $d(i)$ ist die kürzeste Weglänge von s nach i , $R(i)$ ist der direkte Vorgänger von i auf einem kürzestem Weg von s nach i .

Wiederholung

Beispiele für
Netzwerkmodelle
basierend auf
GraphenModellierung mittels
GraphenBerechnung kürzester
Wege in Graphen

Problem

Funktionsprinzip

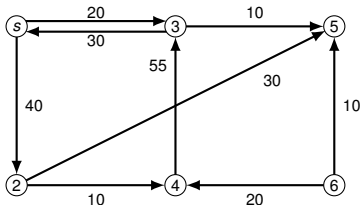
163

Lösung mittels
Dijkstra-AlgorithmusLösung mittels
Simplex-AlgorithmusAnwendungsmöglichkeit
kürzester WegeZusammenfassung &
Ausblick

Beispiel Dijkstra-Algorithmus 1/8

Pseudocode: Zeile Input

Input: gerichteter Graph $G = (V, E, c)$ mit $c_{ij} \geq 0, \forall (i, j) \in E$
Der Startknoten sei s .



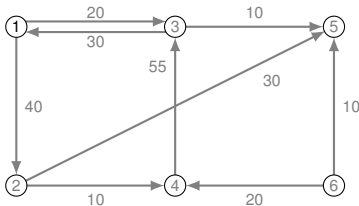
Gesucht: Kürzeste Wege von Knoten s zu allen Knoten in $V \setminus \{s\}$.

Beispiel Dijkstra-Algorithmus 2/8

Pseudocode: Zeile 2 (Initialisierung)

Startknoten $s = 1 \Rightarrow \mathcal{M} = \{1\}, d(1) = 0$

Für alle $i \in V \setminus \{1\}$, setze $d(i) = +\infty$.



i	1	2	3	4	5	6
$d(i)$	0	∞	∞	∞	∞	∞

$\mathcal{M} = \{1\}$

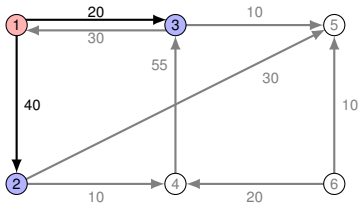
Beispiel Dijkstra-Algorithmus 3/8

Pseudocode: Zeilen 3 bis 11 (1. Iteration, markierter Knoten $h = 1$)

Zeile 4: Welcher Knoten h aus $\mathcal{M} = \{1\}$ soll rot markiert werden? $h = 1$

Zeile 5: Entferne $h = 1$ aus \mathcal{M} , h wird **markiert**, $\mathcal{M} = \emptyset$

Zeilen 6–10: Betrachte alle direkten Nachfolger von $h = 1$ (hier Knoten 2, 3)



i	1	2	3	4	5	6
$d(i)$	0	40	20	∞	∞	∞
$R(i)$		1	1			

$\mathcal{M} = \{2, 3\}$

Von Zeile 11 zurück zu Zeile 3: Beginne neue Iteration, falls \mathcal{M} nichtleer

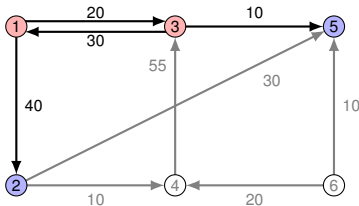
Beispiel Dijkstra-Algorithmus 4/8

Pseudocode: Zeilen 3 bis 11 (2. Iteration, markierter Knoten $h = 3$)

Zeile 4: Wähle zu markierenden Knoten h aus $\mathcal{M} = \{2, 3\}$

Allgemein: $h = \operatorname{argmin}_{i \in \mathcal{M}} d(i)$

Im Beispiel: $3 = \operatorname{argmin}_{i \in \{2, 3\}} \{d(2), d(3)\}$

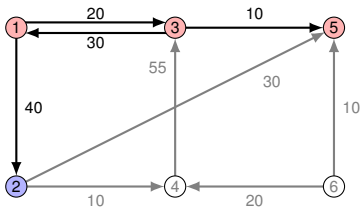


i	1	2	3	4	5	6
$d(i)$	0	40	20	∞	30	∞
$R(i)$		1	1		3	

$$\mathcal{M} = \{2, 5\}$$

Beispiel Dijkstra-Algorithmus 5/8

Pseudocode: Zeilen 3 bis 11 (3. Iteration, markierter Knoten $h = 5$)

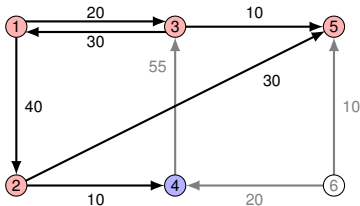


i	1	2	3	4	5	6
$d(i)$	0	40	20	∞	30	∞
$R(i)$		1	1		3	

$$\mathcal{M} = \{2\}$$

Beispiel Dijkstra-Algorithmus 6/8

Pseudocode: Zeilen 3 bis 11 (4. Iteration, markierter Knoten $h = 2$)

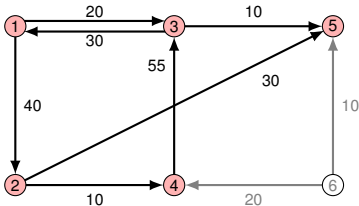


i	1	2	3	4	5	6
$d(i)$	0	40	20	50	30	∞
$R(i)$		1	1	2	3	

$$\mathcal{M} = \{4\}$$

Beispiel Dijkstra-Algorithmus 7/8

Pseudocode: Zeilen 3 bis 11 (5. Iteration, markierter Knoten $h = 4$)



i	1	2	3	4	5	6
$d(i)$	0	40	20	50	30	∞
$R(i)$		1	1	2	3	

$$\mathcal{M} = \{\}$$

Zeile 7: Von Knoten 4 aus gibt es keinen neuen kürzesten Weg zu Knoten 6. \mathcal{M} bleibt daher unverändert.

$\mathcal{M} = \emptyset \rightarrow$ Abbruchbedingung in **Zeile 3** erfüllt, keine weitere Iteration.

Beispiel Dijkstra-Algorithmus 8/8

Pseudocode: Zeile Output – Interpretation Ergebnis

Die Vektoren d und R erhalten nun folgende Werte:

i	1	2	3	4	5	6
$d(i)$	0	40	20	50	30	∞
$R(i)$		1	1	2	3	

 $\mathcal{M} = \emptyset$

Ablezen der kürzesten Weglängen von $s = 1$:

$d(5) = 30 \rightarrow$ kürzeste Distanz von s nach 5 ist 30, etc.

Beachte: $d(6) = \infty \rightarrow$ es existiert kein Weg von $s = 1$ zu Knoten 6.

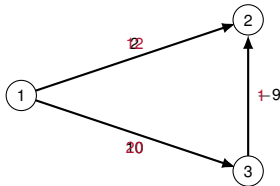
Ablezen des kürzesten Weges von $s = 1$ nach 5:

1. $R(5) = 3$ (Knoten 3 wird unmittelbar vor Knoten 5 besucht)
2. Vorgänger von 3? $R(3) = 1$ (Kn. 1 wird unmittelbar vor Kn. 3 besucht)
3. Vorgänger von 1? $R(1)$ ohne Wert, da Startknoten.

Der **kürzester Weg von 1 nach 5** lautet: 1, 3, 5

Startknoten: $s = 1$

Addiere **10** zu allen Kantengewichten



Kürzester Weg von 1 nach 2: Weg 1, 3, 2 mit Distanz +1

Dijkstra findet: Weg 1,2 mit Distanz +2

Vorsicht: Kantengewichte erhöhen keine Lösung

↪ Unterschiedlich Kantenzahl kürzester Wege

Abhilfe: Bellman-Ford-Algorithmus

Wiederholung

Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

Berechnung kürzester
Wege in Graphen

Problem

Funktionsprinzip

172

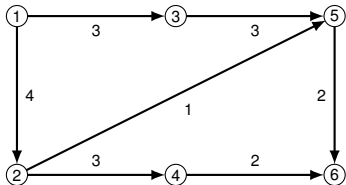
Lösung mittels
Dijkstra-Algorithmus
Lösung mittels
Simplex-Algorithmus

Anwendungsmöglichkeit
kürzester Wege

Zusammenfassung &
Ausblick

Idee: $d(i)$ mit Entscheidungsvariable x_i abbilden

$\hookrightarrow x_i$ bezeichnet kürzeste Entfernung Startknotens zu Knoten i



Zielfunktion: $\max x_6 - x_1$
pro Kante (i, j) eine Restriktion

$$-x_1 + x_2 \leq 4 \quad \text{Pfeil (1,2)}$$

$$-x_1 + x_3 \leq 3 \quad \text{Pfeil (1,3)}$$

$$-x_2 + x_4 \leq 3 \quad \text{Pfeil (2,4)}$$

$$-x_2 + x_5 \leq 1 \quad \text{Pfeil (2,5)}$$

$$-x_3 + x_5 \leq 3 \quad \text{Pfeil (3,5)}$$

$$-x_4 + x_6 \leq 2 \quad \text{Pfeil (4,6)}$$

$$-x_5 + x_6 \leq 2 \quad \text{Pfeil (5,6)}$$

$$x_i \geq 0 \quad \text{für alle } i \in V$$

Allgemeines Modell für Digraph

$$\begin{aligned} \max \quad & x_t \\ \text{u.d.N.} \quad & x_j - x_i \leq c_{ij} \quad \forall (i, j) \in E \\ & x_s = 0 \end{aligned}$$

Beispiele für Netzwerkmodelle basierend auf Graphen

Modellierung mittels Graphen

Grundlegende Strukturen

Repräsentation von Graphen mit Matrizen

Berechnung kürzester Wege in Graphen

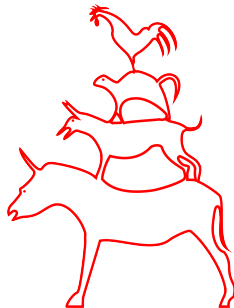
Problem

Funktionsprinzip

Lösung mittels Dijkstra-Algorithmus

Lösung mittels Simplex-Algorithmus

Anwendungsmöglichkeiten kürzester Wege



Operations Research

J. Pannek

Wiederholung

Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

Berechnung kürzester
Wege in Graphen

173 Anwendungsmöglichkeiten
kürzester Wege

Zusammenfassung &
Ausblick

Reiseplanung im Tür-zu-Tür-Service

Minimiere z.B. Umsteigevorgänge, Reisedauer

Wiederholung

Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

Berechnung kürzester
Wege in Graphen

174 Anwendungsmöglichkeiten
kürzester Wege

Zusammenfassung &
Ausblick

Würzburg Hbf Do, 19.11.15 ab 20:30 3:27 0 ICE **67,00 EUR** **106,00 EUR** → Rückfahrt hinzufügen
Bremen Hbf Do, 19.11.15 an 23:57 → Zur Buchung → Zur Buchung

Bahnhof/Haltestelle	Datum	Zeit	Gleis	Produkte
Würzburg Hbf	Do, 19.11.15	ab 20:30	7	ICE 732
Bremen Hbf	Do, 19.11.15	an 23:57	3	Intercity-Express Richtung: Oldenburg(Oldb) Bordrestaurant

Zwischenhalte einblenden Verspätungs-Alarm Merken In Kalender eintragen Druckansicht

fährt nicht täglich, 17. Nov bis 12. Dez 2015 Mo - Sa

Am Bahnhof Karte anzeigen Aktuelle Alternativen

Würzburg Hbf Do, 19.11.15 ab 21:31 4:16 2 ICE, RE **57,00 EUR** **106,00 EUR** → Rückfahrt hinzufügen
Bremen Hbf Fr, 20.11.15 an 01:47 → Zur Buchung → Zur Buchung

Bahnhof/Haltestelle	Datum	Zeit	Gleis	Produkte
Würzburg Hbf	Do, 19.11.15	ab 21:31	6	ICE 580
Fulda	Do, 19.11.15	an 22:02	6	Intercity-Express Richtung: Kassel-Wilhelmshöhe Bordrestaurant
Umsteigezeit 9 Min.				→ Umsteigezeit anpassen
Fulda	Do, 19.11.15	ab 22:11	6	ICE 272
Hannover Hbf	Fr, 20.11.15	an 00:02	8	Intercity-Express Richtung: Hamburg-Altona Bordrestaurant
Umsteigezeit 19 Min.				→ Umsteigezeit anpassen
Hannover Hbf	Fr, 20.11.15	ab 00:21	11	RE 4400
Bremen Hbf	Fr, 20.11.15	an 01:47	6	Regional-Express Richtung: Bremen Hbf Fahrradmitnahme begrenzt möglich , Fahrzeuggebundene Einstiegshilfe vorhanden

Zwischenhalte einblenden Verspätungs-Alarm Merken In Kalender eintragen Druckansicht

fährt nicht täglich, 17. Nov bis 11. Dez 2015 Mo - Fr

Am Bahnhof Karte anzeigen Aktuelle Alternativen



Wiederholung

Beispiele für
Netzwerkmodelle
basierend auf
Graphen

Modellierung mittels
Graphen

Berechnung kürzester
Wege in Graphen

175 Anwendungsmöglichkeiten
kürzester Wege

Zusammenfassung &
Ausblick

Fragen zur Wiederholung:

- ▶ Was bedeutet die Begriffe Digraph, Weg, Adjazenzmatrix?
- ▶ Wie funktioniert das Dijkstra-Verfahren?

Klausurrelevante Literatur

Domschke/Drexl Kapitel 3.1 und 3.2, oder Nickel/Stein/Waldman
Kapitel 3.1 und 3.2, oder Werners Kapitel 5

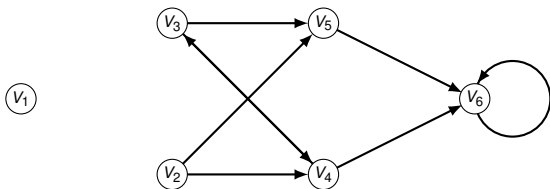
Ausblick

- ▶ Tutorium: Dijkstra, Graphentheoretische Begriffe
- ▶ Vorlesung: MaximalfluSS-Problem, Problem des kostenminimalen FluSSes sowie Lösungsverfahren, Suhl/Mellouli Kapitel 6.8

Teil VI

Graphentheorie II – Netzflussprobleme

Beispiel gerichteter Graph:



Knotenmenge

$$V_2 = \{1, 2, 3, 4, 5, 6\}$$

Kantenmenge

$$E_2 = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$$

Inzidenzabbildung

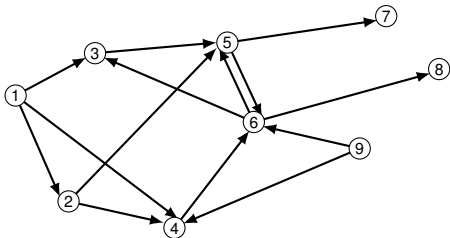
$$\begin{aligned} \omega: e_1 &\rightarrow (2, 4), e_2 \rightarrow (2, 5), \\ e_3 &\rightarrow (3, 4), e_4 \rightarrow (3, 5), \\ e_5 &\rightarrow (4, 3), e_6 \rightarrow (4, 6), \\ e_7 &\rightarrow (5, 6), e_8 \rightarrow (5, 6) \end{aligned}$$

Nachbarn

$$\begin{aligned} N(1) &= \emptyset, N(2) = \{4, 5\}, \\ N(3) &= \{4, 5\}, N(4) = \{2, 3, 5\}, \\ N(5) &= \{2, 3, 6\}, N(6) = \{4, 5, 6\} \end{aligned}$$

Minimalität

Ein minimaler Weg besteht aus minimalen Teilwegen.



Annahme: minimaler 1-5-Weg ist 1, 2, 4, 6, 5

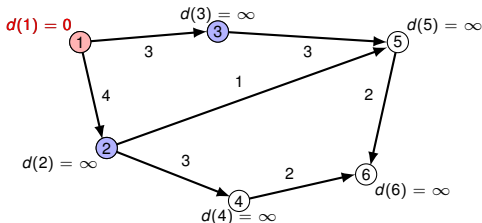
Folgerung: minimaler 1-6-Weg ist 1, 2, 4, 6

Dreiecksungleichung

Falls d_{uv} die kürzeste Distanz zwischen Knoten u und v ist, dann gilt für einen beliebigen Knoten x

$$d_{uv} \leq d_{ux} + d_{xv}.$$

Gegeben: Gerichteter Graph $G = (V, E, c)$ mit **nichtnegativen** Kantengewichten und einem Startknoten, z.B. $1 \in V$.

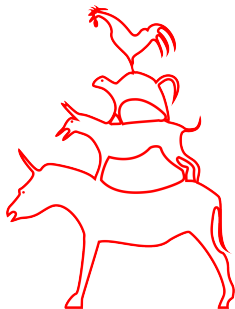


Anmerkung Handout-Version: die relevanten Details sind nur im animierten Foliensatz erkennbar.

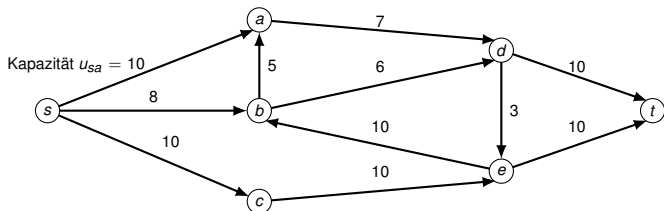
Flüsse in Netzen

Max-Flow-Problem
Problemstellung
Verfahren

Min-Cost-Flow-Problem
Problemstellung
Verfahren



- ▶ Ein Digraph $G = (V, E)$
- ▶ Zwei besondere Knoten s (Quelle) und t (Senke)
- ▶ **Quelle**: ein Knoten **ohne Vorgänger** aber mit mind. einem Nachfolger
- ▶ **Senke**: ein Knoten **ohne Nachfolger** aber mit mind. einem Vorgänger
- ▶ Jede Kante $e \in E$ hat eine **Kapazität** $u(e)$ (bzw. $u_{ij} \geq 0, (i, j) \in E$)

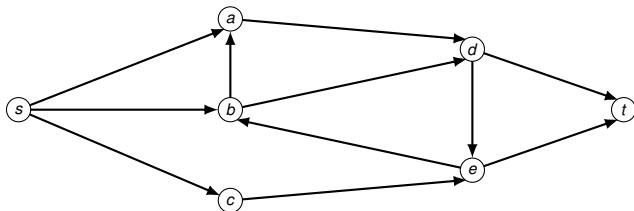


s-t-Fluss

Ein s-t-Fluss (engl. flow) ist eine Funktion, die jeder Kante $e = (i, j) \in E$ einen **Flusswert** $x_{ij} \geq 0$ zuweist und folgendes erfüllt:

- ▶ **Kapazitätskonform**: Der Flusswert einer Kante ist höchstens so groß wie die Kapazität der Kante, d.h. $0 \leq x_{ij} \leq u_{ij}$, $\forall (i, j) \in E$
- ▶ **Flusserhaltung**: In jedem Knoten mit Ausnahme von Quelle s und Senke t muss genau so viel hineinfließen wie herausfließen, d.h.

$$v \in V \setminus \{s, t\} : \sum_{e \text{ nach } v} x_e = \sum_{e \text{ verlässt } v} x_e$$

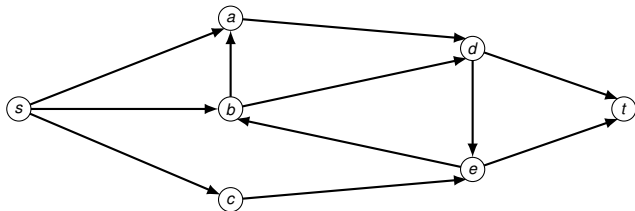


Wert eines Flusses

Der Wert z eines Flusses \mathbf{x} ist

$$z(\mathbf{x}) = \sum_{i \in V} x_{si}.$$

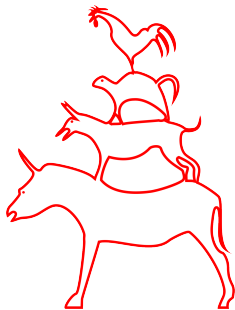
Der Wert entspricht dem (Netto-)Fluss, der aus der Quelle s abfließt.



Flüsse in Netzen

Max-Flow-Problem
Problemstellung
Verfahren

Min-Cost-Flow-Problem
Problemstellung
Verfahren



Maximalflussproblem

Wie viele Flusseinheiten können **maximal** von einem Startknoten zu einem Zielknoten **transportiert werden**?

Anwendung bei **Kapazitätsuntersuchung** in Netzen, z.B.:

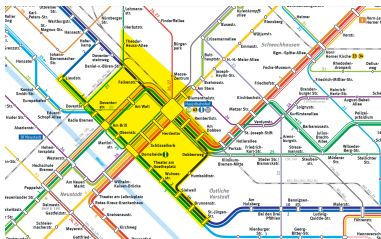
- ▶ Petroleum Produkte in einem Netz aus Pipelines
- ▶ Nachrichten/Datenpakete in einem Telekommunikationsnetz
- ▶ Autos in StraSSennetz

Weitere Anwendungen:

- ▶ Projektauswahl
- ▶ Bildverarbeitung
- ▶ Einsatzplanung von Flugzeugen (Airline Scheduling)
- ▶ ...

Frage

Welche Auswirkung hat eine zusätzliche Baustelle auf den Verkehrsfluss / auf die Kapazität eines ÖPNV Netzes?



Netz des ÖPNV = gerichteter Graph

Knoten = Haltestelle, **Kante** = Straße/Schiene

Gewicht einer Kante = Kapazität des Verkehrsflusses pro Periode

Gegeben

- ▶ Graph $G = (V, E)$ mit Quelle $s \in V$ und Senke $t \in V, s \neq t$
- ▶ Jede Kante $(i, j) \in E$ besitzt eine **maximal zulässige Kapazität** u_{ij}

Gesucht

Maximaler Fluss x in G von s nach t , so dass

- ▶ für alle Knoten mit Ausnahme von s und t die **Flusserhaltungsbedingung** gilt, und
- ▶ die **Kapazitätsbeschränkung** jeder Kante eingehalten wird.

Fluss und Kapazitätsbeschränkung

Für jede Kante $(i, j) \in A$ ist der **Fluss** x_{ij} mit $0 \leq x_{ij} \leq u_{ij}$ festzulegen.

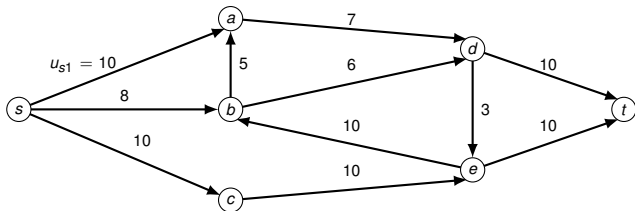
Flusserhaltungsbedingung für einen Knoten i

Die Summe des in einen Knoten i einfließenden Flusses ($inflow(i)$) ist gleich der Summe des aus Knoten i ausfließenden Flusses ($outflow(i)$), d.h.

$$inflow(i) - outflow(i) = 0.$$

Maximaler Fluss

Der **Wert z** eines Flusses x ist das, was an der Senke t ankommt, ohne von dort wieder weggeleitet zu werden. Der maximale Fluss ist der Fluss mit **maximalem Wert**.



Formulierung des Max-Flow-Problems 3/3

Formulierung als Lineares Optimierungsproblem

Gesucht

Maximaler Fluss \mathbf{x} von s nach t , der die **oberen Flussschranken** u_{ij} erfüllt

Entscheidungsvariablen

x_{ij} Flusseinheiten auf der gerichteten Kante (i, j)

Lineares Optimierungsproblem

max \mathbf{x}

$$\text{s.t.} \quad \sum_{j:(i,j) \in E} x_{ij} - \sum_{j:(i,j) \in E} x_{ij} = \begin{cases} \mathbf{x} & \text{für } i = s \\ 0 & \text{für } i \in V \setminus \{s, t\} \\ -\mathbf{x} & \text{für } i = t \end{cases}$$
$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in E$$

Technische Voraussetzung: ein **symmetrisches Flussnetz** (Graph)

Symmetrischer Graph

Ein Graph heißt **symmetrisch**, wenn gilt

$$(i, j) \in E \Leftrightarrow (j, i) \in E,$$

d.h. zwischen zwei Knoten gibt es entweder keine Kante oder aber zwei entgegengesetzte Kanten.

In **asymmetrischen** Graphen ergänze **fehlende Kanten**:

Falls $(i, j) \in E$ und $(j, i) \notin E$, dann füge (j, i) mit $u_{ji} = 0$ in E ein.

Bei Handrechnung kann auf diese Schritte i.d.R. verzichtet werden.

Algorithmus 4: Ford–Fulkerson Algorithmus

Input : Symmetrischer Graph $G = (V, E, c)$ mit Initialfluss \mathbf{x}^0

```
1 begin
2   while  $\exists P$  do
3     |   Identifiziere flussvergrößernden Weg  $P$  von  $s$  nach  $t$ 
4     |   Definiere Hilfsnetz mit Restflusskapazitäten (Residualnetz)
5     |   Vergrößere Fluss  $\mathbf{x}$  um minimale Restflusskapazität entlang  $P$ 
6   end
7 end
```

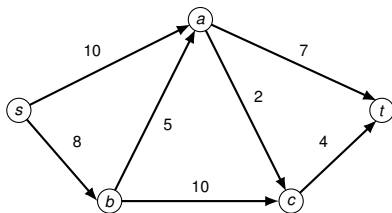
Output: Maximaler Fluss \mathbf{x} von s nach t

Notation: x_{ij} = Fluss von Knoten i zu Knoten j
 x_{ij}^0 = Fluss auf (i, j) im Initialzustand
 x_{ij}^1 = Fluss in erster Iteration
 x_{ij}^k = Fluss in k -ter Iteration.

Was ist ein flussvergrößernder Weg?

Flussvergrößernder Weg

Ein Weg von s nach t heißt **flussvergrößernd**, wenn die Kapazitäten jeder verwendeten Kante positiv (> 0) sind.



Der Weg $P = ((s, a), (a, c), (c, t))$ ist **flussvergrößernd**.

Um wie viel wird durch P der Fluss vergrößert?

Es gilt das Flaschenhals-Prinzip! Auf dem Weg P kann der Fluss um höchstens $\delta = \min\{10, 2, 4\} = 2$ Einheiten erhöht werden.

Wie ist das Residualnetz zu konstruieren?

Residualnetz

Ein Residualnetz $G(x)$ für Fluss x zeigt die noch **unverbrauchten Flusskapazitäten** des ursprünglichen Flussnetzes G an.

Gegeben: Weg P , über den ein zusätzlicher Fluss in Höhe von δ von s nach t im Graph $G = (V, E)$ geschickt wird.

Konstruktion Residualnetz

Für jede Kante $(i, j) \in P$:

- ▶ Der zusätzliche Fluss beansprucht auf Kante (i, j) die Kapazität δ , die **Restkapazität** ist $r_{ij} = u_{ij} - \delta$
- ▶ Kapazitätsverringerung auf (i, j) induziert Kapazitätsvergrößerung für (j, i) (Gegenrichtung / Rückwärtskante): $r_{ji} = u_{ji} + \delta$

Hinweis zur Notation:

u_{ij} = obere Flussschranke von (i, j) im ursprünglichen Flussnetz G

r_{ij} = verbleibende Kapazität im Residualnetz $G(x)$

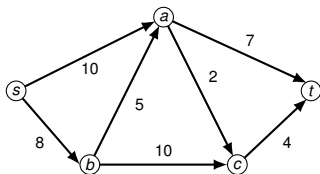
Wie ist das Residualnetz zu konstruieren?

Beispiel

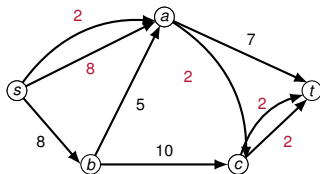
Gegeben: Flussnetz G mit Nullfluss x^0 sowie flussvergrößernder

Weg $P = ((s, a), (a, c), (c, t))$

Flussnetz $G(x^0)$:



Residualnetz $G(x)$:



Fluss x^0 wird um $\delta = \min_{(i,j) \in P} \{10, 2, 4\} = 2$ erhöht.

Restkapazität

Pfeile mit Restkapazität Null werden i.A. nicht gezeichnet.

Zweck Rückwärtskanten

Rückwärtskanten (j, i) erlauben Reduktion auf Kante (i, j) .

→ Entscheidungen reversibel

Algorithmus 5: Augmenting-Path-Algorithmus

Input : Symmetrischer Graph $G = (V, E, c)$ mit Initialfluss \mathbf{x}^0

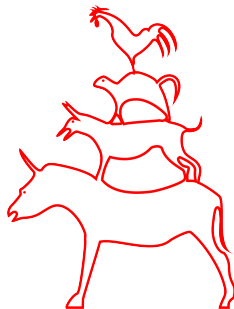
```
1 foreach  $(i, j) \in E$  do
2    $r_{ij} := (u_{ij} - x_{ij}^0) + (x_{ji}^0 - l_{ji})$ 
3 while  $G(x)$  hat gerichteten Weg von  $s$  nach  $t$  mit Kanten  $(i, j) | r_{ij} > 0$  do
4   Identifiziere flussvergrößernden Weg  $P$ 
5   Sei  $\delta := \min_{(i,j) \in P} r_{ij}$ 
6   foreach  $(i, j) \in P$  do
7      $r_{ij} := r_{ij} - \delta$ 
8      $r_{ji} := r_{ji} + \delta$ 
9 foreach  $(i, j) \in E$  do
10  if  $u_{ij} - r_{ij} > l_{ij}$  then
11     $x_{ij} := u_{ij} - r_{ij}$ 
12  else
13     $x_{ij} := l_{ij}$ 
```

Output: Maximaler Fluss \mathbf{x} auf G von s nach t der Grösse
 $z = \text{outflow}(s) - \text{inflow}(s)$

Flüsse in Netzen

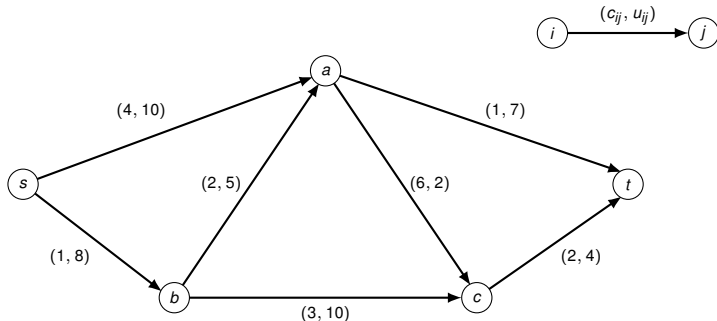
Max-Flow-Problem
Problemstellung
Verfahren

Min-Cost-Flow-Problem
Problemstellung
Verfahren



Unterschied zum Max-Flow-Problem:

- ▶ **Gegeben:** Obere Flussschranken u_{ij} und Kosten $c_{ij} \geq 0$ für jede Kante (i, j)
- ▶ **Gegeben:** Transportmenge b_t (Bedarf Knoten t)
- ▶ **Gesucht:** Kostenminimaler Fluss (von s nach t) der Stärke b_t



Restkapazität

Pfeile mit Restkapazität Null werden i.A. nicht gezeichnet.

SSP funktioniert ähnlich wie das Augmenting-Path-Verfahren für das Max-Flow-Problem.

Algorithmus 6: Ford & Fulkerson-Algorithmus

Input : Symmetrischer Graph $G = (V, E, c)$ mit Initialfluss \mathbf{x}^0 ,
gesuchte Flussstärke b_t

```
1 begin
2   while  $\exists P$  und  $\mathbf{x} < b_t$  do
3     Identifiziere kostenminimalen flussvergrößernden Weg
        $P$  von  $s$  nach  $t$ 
4     Definiere Residualnetz mit Restflusskapazitäten
5     Vergrößere Fluss  $\mathbf{x}$  um minimale Restflusskapazität in  $P$ 
6   end
7 end
```

Output: Maximaler Fluss \mathbf{x} von s nach t

Wie ist das Residualnetz zu konstruieren?

Gegeben: Weg P , über den ein zusätzlicher Fluss in Höhe von δ von s nach t im Graph $G = (V, E)$ geschickt wird.

Konstruktion Residualnetz

Für jede Kante $(i, j) \in P$:

- ▶ Verringerung der Kapazität auf Kante (i, j) um δ : $r_{ij} = u_{ij} - \delta$
- ▶ Kapazitätsverringern auf (i, j) induziert Kapazitätsvergrößerung für Rückwärtskante (j, i) : $r_{ji} = u_{ji} + \delta$
- ▶ **NEU:** Die Kosten c_{ji} auf der Rückwärtskante (j, i) entsprechen den negativen Kosten von (i, j) , d.h.: $c_{ji} = (-1) \cdot c_{ij}$

Zweck Residualnetz

Entscheidungen können bei Bedarf rückgängig gemacht werden.

Hinweis zur Notation:

u_{ij} = obere Flusssschranke

r_{ij} = verbleibende Kapazität im Residualnetz $G(x)$

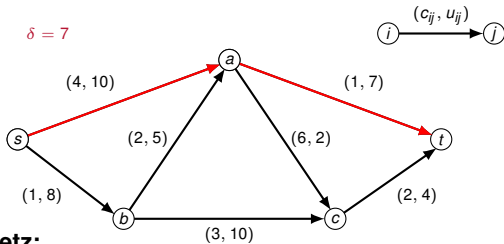
c_{ij} = Kosten für den Transport einer Flusseinheit auf (i, j)

Wie ist das Residualnetz zu konstruieren?

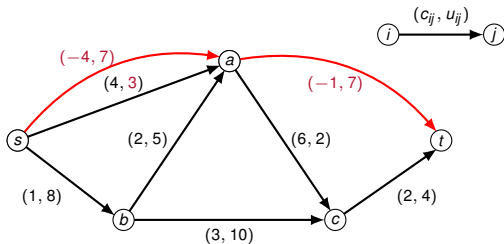
Beispiel

Gegeben: Flussvergrößernder Weg $P = ((s, a), (a, t))$ mit $\delta = 7$ im Flussnetz G

(Achtung: P ist nicht der kostengünstigste Weg!)



Residualnetz:



Algorithmus 7: Successive-Shortest-Path-Algorithmus (SSP)

Input : Symmetrischer Graph $G = (V, E, c)$ mit Initialfluss \mathbf{x}^0

```
1 foreach  $(i, j) \in E$  do
2    $r_{ij} := u_{ij} - x_{ij}^0$ 
3    $r_{ji} := x_{ij}^0 - l_{ij}$ 
4    $c_{ji} := -c_{ij}$ 
5    $\mathbf{x} := \mathbf{x}^0$ 
6 while  $G(x)$  hat gerichteten Weg von  $s$  nach  $t$  mit Kanten  $(i, j) | r_{ij} > 0 \wedge \mathbf{x} < b_t$  do
7   Identifiziere kürzesten flussvergrößernden Weg  $P$ 
8   Sei  $\delta := \min_{(i,j) \in P} r_{ij}$ 
9   if  $\mathbf{x} + \delta > b_t$  then
10     $\delta := b_t - \mathbf{x}$ 
11     $\mathbf{x} := b_t$ 
12  else
13     $\mathbf{x} := \mathbf{x} + \delta$ 
14  foreach  $(i, j) \in P$  do
15     $r_{ij} := r_{ij} - \delta$ 
16     $r_{ji} := r_{ji} + \delta$ 
17 foreach  $(i, j) \in E$  do
18    $x_{ij} := u_{ij} - r_{ij}$ 
```

19 **if** $\mathbf{x} < b_t$ **then** Es existiert kein Fluss der Grösse b_t ; \mathbf{x} ist maximal

20

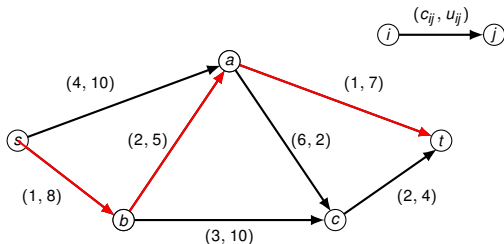
Output: Kostenminimaler Fluss (oder maximaler Fluss) \mathbf{x} auf G von s nach t der Grösse $\mathbf{x} = b_t$

Successive-Shortest-Path-Algorithmus, Zeile 6 bis 8

```
6 while  $G(x)$  hat gerichteten Weg von  $s$  nach  $t$  mit Kanten  $(i, j) | r_{ij} > 0 \wedge x < b_t$  do  
7   Identifiziere kürzesten flussvergrößernden Weg  $P$   
8   Sei  $\delta := \min_{(i,j) \in P} r_{ij}$   
   ...  
...
```

Kürzester flussvergrößernder Weg P über die Pfeile
 (s, b) , (b, a) , (a, t)

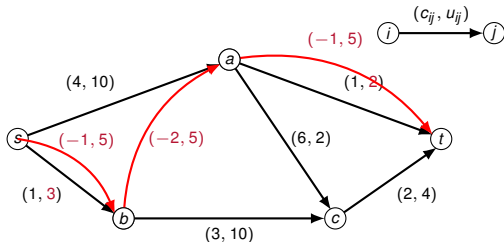
Erhöhung Flusstärke $\delta = \min_{(i,j) \in P} r_{ij} = \min\{8, 5, 7\} = 5$



Successive-Shortest-Path-Algorithmus, Zeile 14 bis 16

```
...  
14 foreach  $(i, j) \in P$  do  
15    $r_{ij} := r_{ij} - \delta$   
16    $r_{ji} := r_{ji} + \delta$   
...
```

$P = ((s, b), (b, a), (a, t))$, Restkapazität $\delta = 5$



Unterschiede bei der Lösung von Max-Flow vs. Lösung von Min-Cost-Flow

Fragestellung

Max-Flow: Wie viel Fluss kann durch ein Netz maximal flieSSen?

Min-Cost-Flow: Welches sind die geringsten Kosten für einen festen Fluss mit Wert b_t durch ein Netz?

Unterschiede bei der Lösung von Min-Cost-Flow zu Max-Flow:

1. Jede Kante ist mit **Kosten und Kapazität** bewertet. Muss im Residualnetz beachtet werden.
2. Statt irgend einen flussvergrößernden Weg zu suchen, ist **nur** der jeweils **kostengünstigste** Weg erlaubt.
3. SSP endet, sobald die geforderte Flussstärke b_t (gegeben!) erreicht ist. Gibt es keinen Fluss mit Stärke b_t , ist das Problem nicht lösbar.

Fragen zur Wiederholung:

- ▶ Was bedeutet die Begriffe Fluss, flussvergrößernder Weg, Residualnetz?
- ▶ Was ist das Max-Flow-Problem?
- ▶ Was ist das Min-Cost-Flow-Problem?
- ▶ Wie funktionieren Augmenting-Path-Verfahren und Successive-Shortest-Path-Verfahren?

Klausurrelevante Literatur

Suhl/Mellouli Kapitel 6.8

Ausblick

- ▶ Tutorium: Max-Flow, Min-Cost-Flow Problem
- ▶ Vorlesung: Varianten des Min-Cost-Flow-Problems bzw. Transshipmentproblems, Suhl/Mellouli Kapitel 6.5 — 6.9 und Anwendung ÖPV Suhl/Mellouli Kapitel 7

Teil VII

Graphentheorie III – Modellierungstechniken für Netzflussprobleme

Ausgewählte Flussprobleme

Das Zuordnungsproblem

Das Transportproblem

Das Umladeproblem

Modellierungstechniken für Flussprobleme

Untere Flussschranken

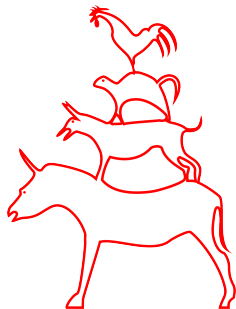
Mehrere Quellen und/oder mehrere Senken

Zirkulationsfluss

Antisymmetrisches Netz

Node Splitting

Fallbeispiel: Umlaufplanung im ÖPV



204 **Ausgewählte
Flussprobleme**

Das Zuordnungsproblem

Das Transportproblem

Das Umladeproblem

Modellierungstechniken
für Flussprobleme

Fallbeispiel:
Umlaufplanung im
ÖPV

Zusammenfassung &
Ausblick

Problem: Schwimmer für die 4x200m Lagenstaffel

Das Schwimmsportteam der Uni nimmt an den Unimeisterschaften in der Disziplin Lagenstaffel (4x200m) teil.

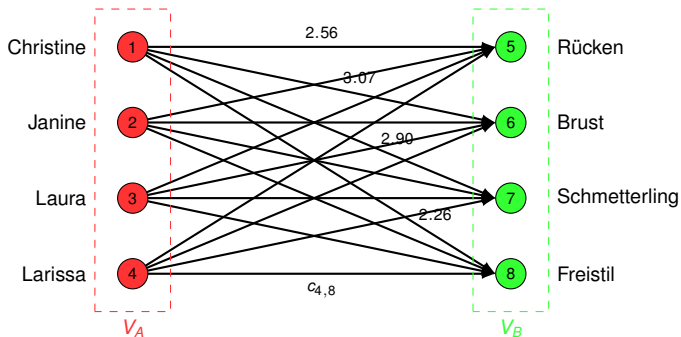
Der Trainerin sind die folgenden Bestzeiten ihres Teams bekannt:

	Zeit (min)			
	Rücken	Brust	Schmetterling	Freistil
Christine	2.56	3.07	2.90	2.26
Janine	2.63	3.01	3.12	2.35
Larissa	2.71	2.95	2.96	2.29
Laura	2.60	2.86	3.08	2.41

Welche Schwimmerin soll in der Lagenstaffel welche Disziplin schwimmen, damit die Siegchance maximal wird?

Das lineare Zuordnungsproblem

Linear Assignment Problem, LAP



Operations Research

J. Pannek

Ausgewählte
Flussprobleme

206

Das Zuordnungsproblem

Das Transportproblem

Das Umladeproblem

Modellierungstechniken
für Flussprobleme

Fallbeispiel:
Umlaufplanung im
ÖPV

Zusammenfassung &
Ausblick

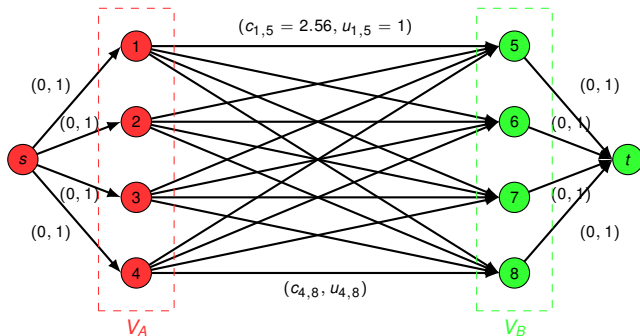
Der gerichtete Graph $G = (V, E, c)$ ist *bipartit*, d.h.

- ▶ die Knotenmenge V ist in zwei Teilmengen V_A und V_B partitioniert ($V = V_A \cup V_B$ mit $V_A \cap V_B = \emptyset$);
- ▶ Kanten von G beginnen stets mit einem Knoten aus V_A und endet mit einem Knoten aus V_B ($E \subseteq V_A \times V_B$).

Beim LAP gilt außerdem: $|V_A| = |V_B|$

Rückführung des LAP auf ein Min-Cost-Flow-Modell:

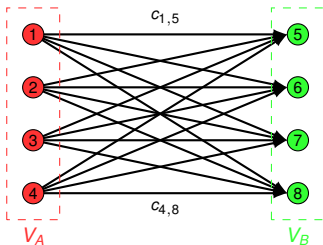
1. Führe **künstliche Quelle und Senke** s und t ein
2. Wähle geeignete Kantengewichte ($c_{si} = c_{it} = 0, i \in V, u_{ij} = 1, \{i, j\} \in V \cup \{s, t\}$)
3. Berechne einen kostenminimalen Fluss mit $b_t = 4$



Das lineare Zuordnungsproblem als Lineares Programm

Matchingproblem – Stable Marriage Problem

Gegeben: bipartiter gerichteter Graph $G = (V, E, c)$.



Gesucht: kostenminimale Zuordnung je eines Elements aus V_A zu genau einem Element aus V_B

i.A. gilt beim Matching: $|V_A| \neq |V_B|$

$$\min \sum_{i \in V_A} \sum_{j \in V_B} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j \in V_B} x_{ij} = 1, \quad \forall i \in V_A$$

$$\sum_{i \in V_A} x_{ij} = 1, \quad \forall j \in V_B$$

$$x_{ij} \geq 0 \quad \forall i, j \in V$$

Zuordnung von

- ▶ Studierenden zu Studienprojekte (Kosten: Präferenz für Projekt)
- ▶ Projektleitern zu Projekten (Kosten: Eignung für Projekt)
- ▶ Key-Account-Kundenbetreuer zu Kunde
- ▶ Maschinen zu Produktionsaufträgen (Kosten: Arbeitsstunden)

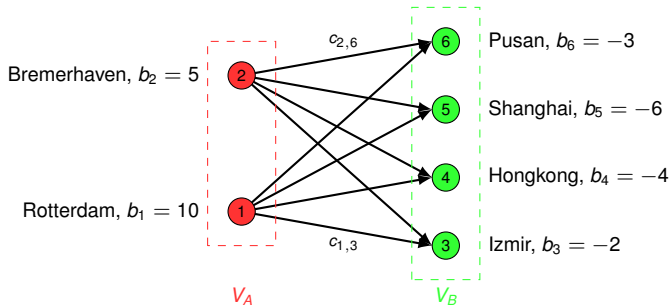
Problem: Repositionierung leerer Container

Transportproblem TPP

Hintergrund: Reederei müssen regelmäßig ihre **leeren Container** zwischen Häfen repositionieren.

Angebotsknoten ($b_i > 0, i \in V_A$) \longleftrightarrow **Nachfrageknoten** ($b_i < 0, i \in V_B$)
Kantengewicht = Transportkosten / Opportunitätskosten

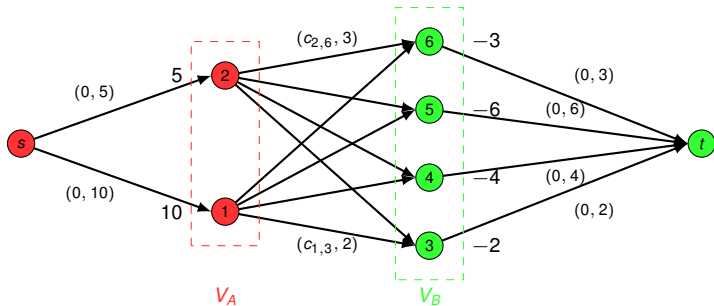
Der Bedarf eines Hafens kann durch **mehrere Häfen** gedeckt werden.



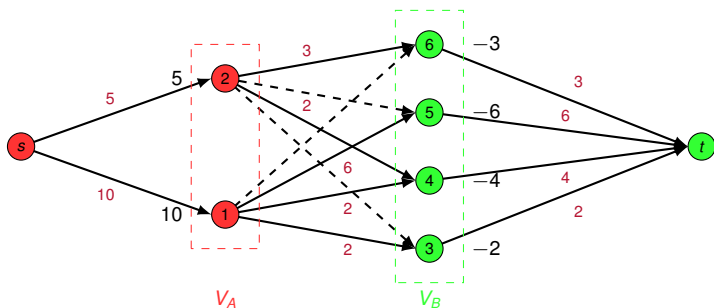
Gesucht: kostengünstigster Transportplan für Nachfrage

Rückführung des TPP auf ein Min-Cost-Flow Modell:

1. Führe **künstliche Quelle und Senke** s und t ein
2. Wähle geeignete Kantengewichte ($c_{si} = c_{it} = 0, i \in V$,
 $u_{si} = b_i, i \in V_A$ und $u_{it} = |b_i|, i \in V_B$) und $u_{ij} = |b_j|, i \in V_A, j \in V_B$
3. Berechne einen kostenminimalen Fluss mit $b_t = \sum_{i \in V_B} |b_i| = 15$



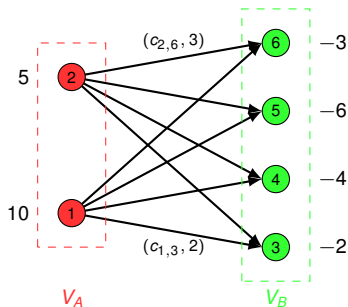
Sei die kostenminimale Min-Cost-Flow Lösung z.B.:



Interpretation Lösung: $x_{15} = 6 \rightarrow$ Transport von 6 leeren Container von Rotterdam (Hafen 1) nach Shanghai (Hafen 5), ...
 x_{si} und x_{jt} ohne besondere Bedeutung für reales Problem.

Kosten Transportplan: $2c_{13} + 2c_{14} + 6c_{15} + 3c_{26} + 2c_{24}$

Gegeben: bipartitierter gerichteter Graph $G = (V, E, c)$.



Gesucht: kostenminimaler Transportplan für Nachfrage $b_j, j \in V_B$

$$\min \sum_{i \in V_A} \sum_{j \in V_B} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j \in V_B} x_{ij} = b_i, \quad \forall i \in V_A$$

$$\sum_{i \in V_A} x_{ij} = |b_j|, \quad \forall j \in V_B$$

$$x_{ij} \geq 0 \quad \forall i, j \in V$$

Erweiterung des TPP:

- ▶ nur indirekte Verbindungen zwischen Angebots- und Nachfrageknoten über Umladeknoten
- ▶ $V = V_A \cup V_B \cup V_T$
- ▶ Bedarf der Umladeknoten ist Null

Anwendungen:

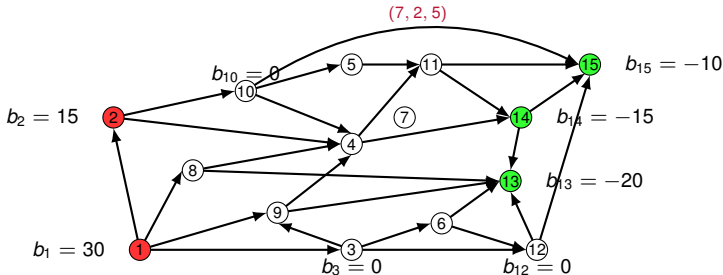
- ▶ Einstufiges TPP: Transport der Güter von Fabrik zu Kunde
- ▶ Zweistufig TPP: Transport der Güter von Fabrik zu Distributionszentrum zu Kunde
- ▶ Zwei- und mehrstufige TPP: Intermodale Transporte

NEU: untere Flussschranke (FS) l_{ij}

Kantenbewertung: $(c_{ij}, l_{ij}, u_{ij}) \sim$ (**Kosten, untere FS, obere FS**).

NEU: Umladeknoten ($b_i = 0$), neben Angebots- bzw.

Nachfrageknoten ($b_i > 0$ bzw. $b_i < 0$).



Annahme: Gesamtangebot = Gesamtnachfrage, d.h. $\sum_{i \in V} b_i = 0$

Umladeproblem

Das Umladeproblem ist gegeben durch

$$\min z = \sum_{(i,j) \in E} c_{ij} x_{ij} \quad (\text{Gesamtkosten des Flusses } x)$$

$$\text{s.t.} \quad \text{outflow}(i) - \text{inflow}(i) = b_i \quad \forall i \in V \quad (\text{Flusserhaltung})$$

$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in E \quad (\text{Flussschranken})$$

mit

$$\text{outflow}(i) := \sum_{j: (i,j) \in E} x_{ij} \quad \text{inflow}(i) := \sum_{j: (j,i) \in E} x_{ji}$$

Falls $b_i > 0$, dann ist i ein **Angebotsknoten** und $\text{outflow}(i) > \text{inflow}(i)$

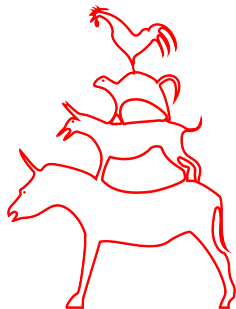
Falls $b_i < 0$, dann ist i ein **Nachfrageknoten** und $\text{outflow}(i) < \text{inflow}(i)$

Falls $b_i = 0$, dann ist i ein **Umladeknoten** und $\text{outflow}(i) = \text{inflow}(i)$

Ausgewählte Flussprobleme
Das Zuordnungsproblem
Das Transportproblem
Das Umladeproblem

Modellierungstechniken für Flussprobleme
Untere Flussschranken
Mehrere Quellen und/oder mehrere Senken
Zirkulationsfluss
Antisymmetrisches Netz
Node Splitting

Fallbeispiel: Umlaufplanung im ÖPV



Untere Flussschranken
Mehrere Quellen und/oder
mehrere Senken
Zirkulationsfluss
Antisymmetrisches Netz
Node Splitting

Fallbeispiel:
Umlaufplanung im
ÖPV

Zusammenfassung &
Ausblick

Generisches Flussmodell meist einfach aufstellbar

Aber: Lösung des generischen Flussmodells (häufig) schwierig

Daher:

- ▶ **Umformung** generisches Flussmodell in
 - ▶ Min-Cost-Flow-Modell
 - ▶ Max-Flow-Modell
- ▶ Nutzung von **Lösungsverfahren** wie Successive-Shortest-Path

Notation:

- ▶ x_{ij} Fluss (in Flusseinheiten, FE) auf gerichteter Kante (i, j)
- ▶ c_{ij} Kosten je FE auf Kante (i, j)
- ▶ l_{ij} und u_{ij} sind untere (lower) und obere (upper) Flusschranken auf Kante (i, j) (in FE)
- ▶ b_i Angebots- ($b_i > 0$) bzw. Nachfragemenge in FE ($b_i < 0$) von Knoten i

Operations Research

J. Pannek

Ausgewählte
Flussprobleme

217 **Modellierungstechniken
für Flussprobleme**

Untere Flusschranken
Mehrere Quellen und/oder
mehrere Senken
Zirkulationsfluss
Antisymmetrisches Netz
Node Splitting

Fallbeispiel:
Umlaufplanung im
ÖPV

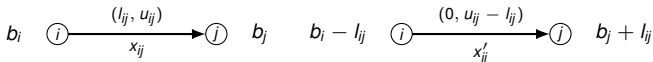
Zusammenfassung &
Ausblick

Graphtransformation für Schranken

Der Graph eines Flussmodells lässt sich so transformieren, dass **positive untere Flussschranken** ($l_{ij} > 0$) entfallen, d.h. alle $l_{ij} = 0$.

Idee: Für jede Kante mit $l_{ij} > 0$

- ▶ Setze $x'_{ij} := x_{ij} - l_{ij}$
- ▶ Wandle $l_{ij} \leq x_{ij} \leq u_{ij}$ um in $0 \leq x'_{ij} \leq u_{ij} - l_{ij}$
- ▶ Einsetzen in Flusserhaltungsbedingungen: $b'_i = b_i - l_{ij}$ bzw. $b'_j = b_j + l_{ij}$



x'_{ij} misst den über l_{ij} hinausgehenden Fluss.

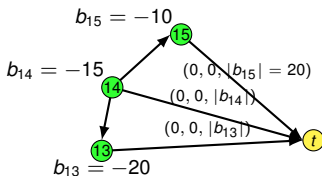
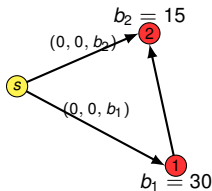
Graphtransformation für Quellen/Senken

Ein Graph mit mehreren Quellen und/oder Senken kann in einen Graphen mit nur einer Quelle s und Senke t transformiert werden.

Idee: Einführung einer **Superquelle s** und **Supersenke t**

Kante von s zu Angebotsknoten i mit Gewicht $(0, 0, b_i)$

Kante von Nachfrageknoten i zu Supersenke t mit Gewicht $(0, 0, |b_i|)$

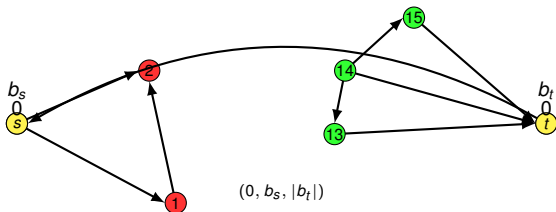


$(c_{ij}, l_{ij}, u_{ij}) = (\text{Kosten, untere, obere Flusschranke})$

Graphtransformation für Zirkulationsfluss

Sollen alle Quellen und Senken eliminiert werden, entsteht ein Zirkulationsflussproblem; hier gibt es nur noch Umladeknoten.

Eignet sich zur Modellierung periodischer Sachverhalte.
Ergänze neue Kante von **Senke t** zu **Quelle s**



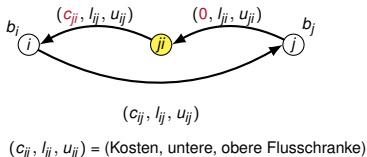
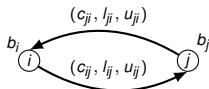
$(c_{ij}, l_{ij}, u_{ij}) = (\text{Kosten, untere, obere Flusschranke})$

Ein zulässiger Fluss hat immer die auf der (t, s) geforderte Stärke b_s
(Stärke Min-Cost-Flow = b_s , Stärke Max-Flow = b_s).

Graphtransformation für Symmetrische Kanten

Symmetrische Flussgraphen (aus $(i, j) \in E$ folgt $(j, i) \in E$) können in **antisymmetrische** Flussgraphen überführt werden (aus $(i, j) \in E$ folgt $(j, i) \notin E$).

Idee: Neuer Knoten ji für jedes Paar $(i, j), (j, i) \in E$.



Kante (i, j) unverändert

Aber Kante (j, i) wird durch (j, ji) und (ji, i) ersetzt.

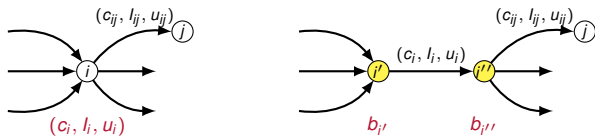
Flusschranken bleiben im Prinzip erhalten, Kosten werden angepasst

Zweck: manchen Algorithmen wird die Arbeit erleichtert

Graphtransformation durch Node-Splitting

Sollen Kosten oder Kapazitätsrestriktionen beim Durchfließen eines **Knoten** statt auf Kanten berücksichtigt werden, dann kann man die Technik des Node-Splitting anwenden.

Idee: Knoten i durch zwei Knoten i' und i'' ersetzen und neue Kante (i' , i'') einfügen



Kosten c_i bzw. Flussschranken l_i , u_i werden auf Kante übertragen

Ausgewählte Flussprobleme

Das Zuordnungsproblem

Das Transportproblem

Das Umladeproblem

Modellierungstechniken für Flussprobleme

Untere Flussschranken

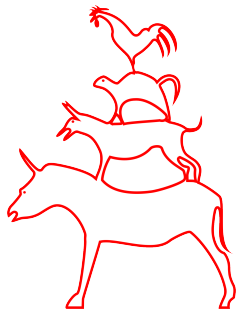
Mehrere Quellen und/oder mehrere Senken

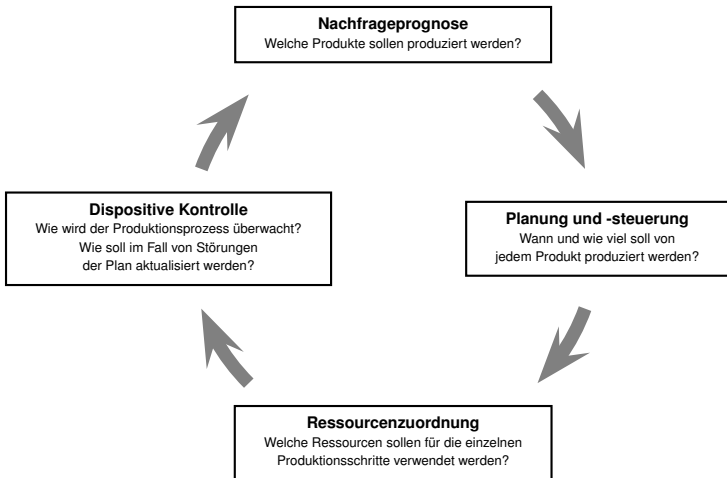
Zirkulationsfluss

Antisymmetrisches Netz

Node Splitting

Fallbeispiel: Umlaufplanung im ÖPV





1. Nachfrageprognose

- ▶ Passagieraufkommen, Auslastung
- ▶ Entwurf Transportnetz: Linien, Kapazitäten, Takt, Saison
- ▶ Produkt = Fahrten/Flüge mit gleicher Linie, Startzeit, Wochentag

2. Planung und -steuerung

- ▶ Flottenzuordnung
- ▶ Umläufe
- ▶ Diensteneinsatzplanung

3. Ressourcenzuordnung

- ▶ konkrete Fahrzeuge/Personen werden Fahrten zugeordnet
- ▶ Fahrzeuge: Wartungsanforderungen
- ▶ Personal: Urlaub, Flugsimulator

4. Dispositive Kontrolle

- ▶ Bewertung Ist-Situation
- ▶ Störungen antizipieren
- ▶ Reaktion auf Störungen

Ein Fahrplan ist gegeben.

Umlauf

Folge von Fahrten (bzw. Flügen), die **zeitlich und räumlich** durch ein Fahrzeug (bzw. Flugzeug) bedient werden können.

Leerfahrt = Bewegung des Transportmittels ohne Fahrgäste

Leerfahrten sind erlaubt und auch sinnvoll

Umlaufplanung

Ausgehend von einem **gegebenen Fahrplan** sind Umläufe für die Fahrzeuge zu bilden, so dass

- ▶ alle Planfahrten bedient werden und
- ▶ die Gesamtkosten (**Fixkosten** Fuhrpark, **variable Kosten** für Leerfahrten) minimiert werden.

Fluss = Fahrzeuge „fließen“ im Netz

Fluss soll **Planfahrten** geeignet „überdecken“

Fahrtverknüpfungen bzw. **Anschlüsse** für Fahrzeuge, d.h.

- ▶ Fahrzeug beendet eine Fahrt, mit welcher Fahrt geht es weiter?
- ▶ Analog zu Passagieranschlüssen

Modellierung:

Planfahrt (Ankunftsereignis, Abfahrtereignis) ~ Knoten

Anschlüsse (Fahrtverknüpfungen) ~ Kanten

Für den folgenden Fahrplan sollen Umläufe geplant werden.

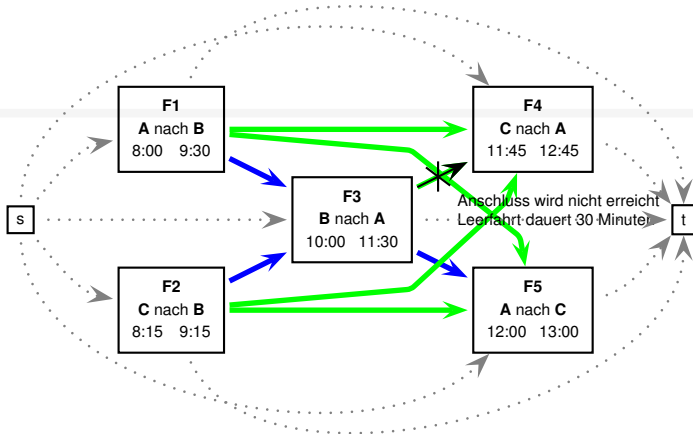
Fahrt	Abfahrt		Ankunft	
	Ort	Zeit	Ort	Zeit
F1	A	8:00	B	9:30
F2	C	8:15	B	9:15
F3	B	10:00	A	11:30
F4	C	11:45	A	12:45
F5	A	12:00	C	13:00

Dauer Leerfahrten (in Minuten) zwischen den Orten:

	A	B	C
A	–	30	30
B		–	20
C			–

- Anschluss an gleicher Station
- Anschluss an verschiedenen Stationen



Fahrt	Abfahrt		Ankunft	
	Ort	Zeit	Ort	Zeit
F1	A	8:00	B	9:30
F2	C	8:15	B	9:15
F3	B	10:00	A	11:30
F4	C	11:45	A	12:45
F5	A	12:00	C	13:00

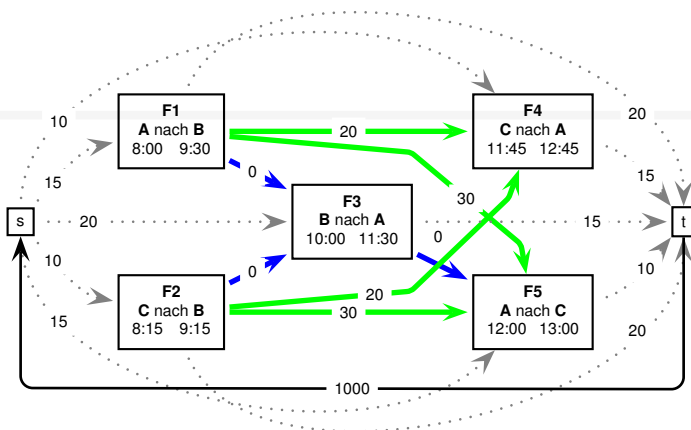


Erläuterung zur vorherigen Folie:

- ▶ Knoten entsprechen (Plan-)Fahrten F_1, F_2, F_3, F_4
- ▶ Kante (i, j) : **Fahrtverknüpfung** möglich, d.h. Fahrt j kann mit dem gleichen Fahrzeug durchgeführt werden wie Fahrt i ;
- ▶ **blau Kante** (i, j) : Ankunftsort von i identisch mit Abfahrtsort von j , **keine Leerfahrt nötig**, z.B. (F_3, F_5)
- ▶ **grüne Kante** (i, j) : Ankunfts- und Abfahrtsort der Planfahrten i und j verschieden, **Leerfahrt nötig**, z.B. (F_1, F_4)
- ▶ **graue Kante**: von Fahrzeugdepot (Quelle t) zu Planfahrt bzw. von Planfahrt zu Fahrzeugdepot (Senke s)
- ▶ keine Kante, z.B. von F_3 nach F_4 , weil F_4 nicht rechtzeitig erreicht wird.

Fallbeispiel 4/6 – Erweiterung um Kosten (variable, fix)

-  Anschluss an gleicher Station
-  Anschluss an verschiedenen Stationen

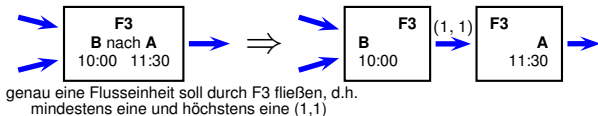


Erläuterung zur vorherigen Folie:

- ▶ Kilometerabhängige (variable) Kosten auf Kanten
- ▶ Fixkosten für Fahrzeugeinsatz, verschiedene Möglichkeiten:
 - a) Depot-Planfahrt auf (s, i) und (i, t) Kanten **oder**
 - b) Kante von Senke t nach Quelle s mit Fixkosten pro Fahrzeug als Kantengewicht $c_{ts} \rightarrow$ Zirkulationsfluss;
- ▶ Jede Flusseinheit muss durch (t, s) Kante fließen. Warum? Das Netz ohne (t, s) ist in jedem Fall **azyklisch** wegen Abfahrzeiten
- ▶ Flusswert auf (t, s) -Kante = Anzahl benötigter Fahrzeuge

Noch liegt kein klassisches Min-Cost-Flow Modell vor. Warum?

- ▶ Fluss muss über jeden Planfahrtnknoten fließen, d.h. Flussschranke auf Knoten und nicht wie erforderlich auf Kante
- ▶ Anwendung der Modellierungstechnik **Node-Splitting**

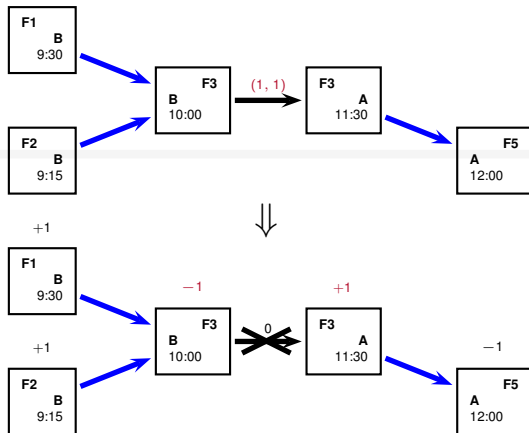


Nun ist Lösung mittels Successive-Shortest-Path-Verfahren möglich

Fallbeispiel – Wer noch nicht genug hat...

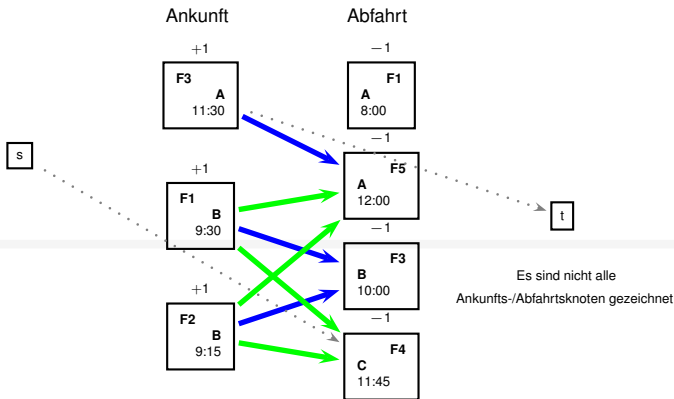
Vom Min-Cost-Flow-Modell zum Quasi-Assignment-Modell

Eliminiere untere Flusschranke auf allen Node-Splitting-Kanten:



Fallbeispiel – Wer noch nicht genug hat...

Vom Min-Cost-Flow-Modell zum Quasi-Assignment-Modell



Jeder Ankunfts-knoten ist mit der Senke t verbunden.

Die Quelle s ist mit jedem Abfahrts-knoten verbunden.

Das Problem entspricht „quasi“ einem Zuordnungsproblem, es heißt daher in der Literatur Quasi-Assignment-Problem.

Fragen zur Wiederholung:

- ▶ Was bedeutet die Begriffe Zuordnungs-, Transport-, und Umladeproblem?
- ▶ Wie funktionieren untere Flussschranken?
- ▶ Wozu dient Node-Splitting?

Klausurrelevante Literatur

Suhl/Mellouli Kapitel 6.5 — 6.9 und Suhl/Mellouli Kapitel 7

Ausblick

- ▶ Tutorium: Max-Flow, Min-Cost-Flow Problem
- ▶ Vorlesung: Modellierung **ganzzahliger** Optimierungsprobleme
Domschke/Drexler Kapitel 6.1, 6.2, 6.5 oder
Nickel/Stein/Waldman Kapitel 5.1, 5.2 oder Suhl/Mellouli Kapitel 4
(ohne 4.6, 4.8)

Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

Zusammenfassung & Ausblick

Teil VIII

Ganzzahlige und kombinatorische Optimierung I – Modellierung

236 **Wiederholung**

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

Zusammenfassung & Ausblick

Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme
Packen, Zerlegen und Überdecken von Mengen
Traveling-Salesman-Problem



Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

Zusammenfassung & Ausblick

Grundstrukturen von Graphen

- ▶ Knoten, Kanten, Pfeile
- ▶ Nachbar-/Vorgänger-/Nachfolger-Beziehung
- ▶ Digraphen, Wege

Standardprobleme auf Graphen

- ▶ Lineares Zuordnungsproblem
- ▶ Transportproblem
- ▶ Umladeproblem
- ▶ Maximalfluss-Problem
- ▶ Kostenminimaler-Fluss-Problem (Min-Cost-Flow)

Netzpläne

- ▶ Vorgangspfeilnetzpläne
- ▶ Vorgangsknotennetzpläne
- ▶ Kostenplanung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

Zusammenfassung & Ausblick

- ▶ Dijkstra-Verfahren zur Lösung des Kürzeste-Wege-Problems
- ▶ Augmenting-Path Verfahren von Ford & Fulkerson für das Maximalfluss-Problem
- ▶ Successive-Shortest-Path-Algorithmus für das Min-Cost-Flow-Problem

239 **Wiederholung**

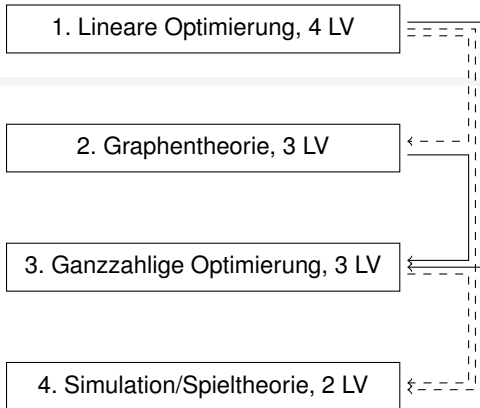
Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

Zusammenfassung & Ausblick



Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme
Packen, Zerlegen und Überdecken von Mengen
Traveling-Salesman-Problem



Ein Dieb auf Beutezug – Das Rucksackproblem 1/2



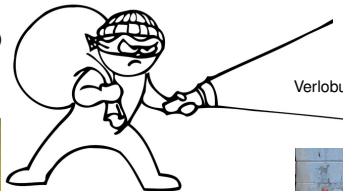
Rolex (0.7kg, 100 T€)



Goldbarren (1 kg, 160 T€)



Verlobungsringe (0.05kg, 30 T€)



Rucksack, max 4 kg



Bild von Picasso (2.4kg, 700 T€)



Bild von Banksy (3kg, 500 T€)

Stehle die Gegenstände, die den Gesamtwert der Beute maximieren, ohne dass die Kapazität des Rucksacks von 4kg überschritten wird.

$$\max \quad 160x_{\text{Gold}} + 30x_{\text{Ring}} + 500x_{\text{Banksy}} + 1000x_{\text{Picasso}} + 100x_{\text{Rolex}}$$

$$\text{u.d.N.} \quad 1x_{\text{Gold}} + 0.05x_{\text{Ring}} + 3x_{\text{Banksy}} + 2.4x_{\text{Picasso}} + 0.7x_{\text{Rolex}} \leq 4 \quad (\text{Kapazität})$$

$$x_i \in \{0, 1\} \quad \text{für } i \in \{\text{Gold, Ring, Banksy, Picasso, Rolex}\}$$

Die Entscheidungsvariablen x_i sind im Rucksackproblem **binär**:

$$x_i = \begin{cases} 1, & \text{stehle Gegenstand } i, \\ 0, & \text{stehle Gegenstand } i \text{ **nicht**.} \end{cases}$$

Das Rucksackproblem - engl. Knapsackproblem

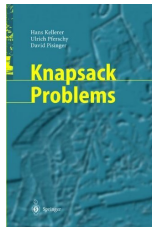
Aus n Objekten mit **Gewichten** $w_i \geq 0$ ($i = 1, \dots, n$) und **Nutzenwerten** $u_i \geq 0$ ($i = 1, \dots, n$) soll eine **nutzenmaximale Auswahl** getroffen werden, so dass eine **maximale Gesamtkapazität** C nicht überschritten wird.

Anwendung: Auswahl Einheiten bei begrenzter Kapazität

$$\max \sum_{i=1}^n u_i x_i$$

$$\text{s.t.} \quad \sum_{i=1}^n w_i x_i \leq C$$

$$x_1, \dots, x_n \in \{0, 1\}$$



Bedeutung: In vielen umfangreicheren Problemen als Teilproblem enthalten

Kellerer, Pferschy, Pisinger
546 Seiten

Bislang: Lineare Optimierungsmodelle (LP) mit **kontinuierlichem** Wertebereich **aller** Entscheidungsvariablen, meist $x \in \mathbb{R}^+$

Erweiterung:

Ganzzahliges Optimierungsproblem

Hat mindestens eine Entscheidungsvariable y einen **diskreten** Wertebereich, z.B. $y \in \mathbb{N}_0$, dann liegt ein **ganzzahliges** (lineares) **Optimierungsmodell (integer program, IP)** vor.

- Gemischt-ganzzahlige** Optimierungsmodelle besitzen diskrete **und** kontinuierliche Variablen (**mixed integer programs, MIP**)
- Kombinatorische Optimierungsmodelle** besitzen ausschließlich Variablen mit **diskreten** Wertebereichen, d.h. es gibt nur **endlich** viele zulässige Lösungen.

Wann sind ganzzahlige Entscheidungsvariablen nötig?

- ▶ Betrachtete Größen sind **unteilbar**
z.B. Produktion von 3.3 Flugzeugen oder Behandlung von 20.7 Patienten kaum sinnvoll
- ▶ Modellierung **logischer Abhängigkeiten**
z.B. *wenn ... , dann ...* Zusammenhänge
- ▶ **Linearisierung** nicht-linearer Modelle
z.B. bei sprungfixen Kosten, alternativen Restriktionen, Skaleneffekten

Hintergrund

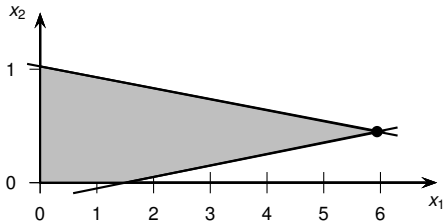
Ganzzahliger Variablen erlauben umfangreichere und realitätsnähere Modellierung

Beispiel ganzzahliges Problem:

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{s.t.} \quad & 0.011x_1 + x_2 \leq 1.01 \\ & -0.011x_1 + x_2 \leq -0.021 \end{aligned}$$

$$x_1, x_2 \in \mathbb{N}_0$$

Wie lautet eine optimale Lösung?



Ganzzahlige Optimierungsmodelle können im Allgemeinen nicht mit dem Simplex-Algorithmus gelöst werden.

Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme
Packen, Zerlegen und Überdecken von Mengen
Traveling-Salesman-Problem



Operations Research

J. Pannek

Wiederholung

Ganzzahlige Entscheidungsvariablen

245 Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

Zusammenfassung & Ausblick

Logische Aussagen

Aussagen sind Sätze, die einen Sachverhalt beschreiben, dem man einen Wahrheitswert (wahr/falsch) zuordnen kann.

„In Bremen schneit es gerade.“ – Aussage kann **wahr** oder **falsch** sein.

„Schneit es in Bremen?“ – Frage, keine Aussage. Nicht beurteilbar, ob Satz selbst wahr oder falsch ist .

Mathematisch lassen sich Aussagen mit **binären** Variablen modellieren:

$$x_i = \begin{cases} 1, & \text{falls die Aussage } i \text{ wahr ist,} \\ 0 & \text{andernfalls.} \end{cases}$$

x_i ist Entscheidungsvariable, die nur zwei Werte annehmen kann.

Im Rucksackproblem des Diebs können etwa Sachverhalte als zusätzliche Restriktionen modelliert werden (**alle x_i binär**):

1. Stehle höchstens 3 Gegenstände:

$$x_1 + x_2 + x_3 + x_4 + x_5 \leq 3 \quad (1)$$

2. Stehle den Ring **und** die Rolex:

$$x_{\text{Ring}} + x_{\text{Rolex}} = 2 \quad (2)$$

3. Stehle den Picasso **und/oder** die Rolex (mind. einen):

$$x_{\text{Picasso}} + x_{\text{Rolex}} \geq 1 \quad (3)$$

4. Stehle **weder** das Gold **noch** den Ring:

$$x_{\text{Gold}} + x_{\text{Ring}} = 0 \quad (4)$$

5. **Wenn** das Bild von Banksy gestohlen wird, **dann** stehle auch das Bild von Picasso:

$$x_{\text{Picasso}} \geq x_{\text{Banksy}} \quad \text{bzw.} \quad x_{\text{Picasso}} - x_{\text{Banksy}} \geq 0 \quad (5)$$

Fall 1: Banksy gestohlen ($x_{\text{Banksy}} = 1$), dann $x_{\text{Picasso}} - 1 \geq 0$.

Um die Restriktion zu erfüllen, muss $x_{\text{Picasso}} = 1$ gelten.

Fall 2: Banksy **nicht** gestohlen ($x_{\text{Banksy}} = 0$), dann $x_{\text{Picasso}} - 0 \geq 0$

Die Restriktion ist sowohl für $x_{\text{Picasso}} = 0$ als auch für $x_{\text{Picasso}} = 1$ erfüllt.

6. Stehle das Gold **dann und nur dann**, wenn auch der Ring gestohlen wird:

$$x_{\text{Gold}} = x_{\text{Ring}} \quad \text{bzw.} \quad x_{\text{Gold}} - x_{\text{Ring}} = 0 \quad (6)$$

Fall 1: Gold gestohlen ($x_{\text{Gold}} = 1$), dann $1 - x_{\text{Ring}} = 0$.

Um die Restriktion zu erfüllen, muss $x_{\text{Ring}} = 1$ gelten.

Fall 2: $x_{\text{Gold}} = 0$, dann $0 - x_{\text{Ring}} = 0$, d.h. $x_{\text{Ring}} = 0$.

7. Die Rolex wird **genau dann** gestohlen, wenn der Ring **nicht** gestohlen wird:

$$x_{\text{Rolex}} = 1 - x_{\text{Ring}} \quad \text{bzw.} \quad x_{\text{Rolex}} + x_{\text{Ring}} = 1 \quad (7)$$

Fall 1: $x_{\text{Ring}} = 1$, dann $x_{\text{Rolex}} + 1 = 1$, d.h. $x_{\text{Rolex}} = 0$.

Fall 2: $x_{\text{Ring}} = 0$, dann $x_{\text{Rolex}} + 0 = 1$, d.h. $x_{\text{Rolex}} = 1$.

Die Entscheidungsvariable x_i wird auch als **binäre Indikatorvariable** bezeichnet.

$$x_i = \begin{cases} 1, & \text{falls Aussage } A_i \text{ wahr ist} \\ 0, & \text{andernfalls.} \end{cases} \quad i = 1, 2$$

Aussage	Logische Verknüpfung
A_1 und/oder A_2 ist wahr	$x_1 + x_2 \geq 1$
A_1 und A_2 sind wahr	$x_1 + x_2 = 2$
Weder A_1 noch A_2 sind wahr	$x_1 + x_2 = 0$
Wenn A_1 wahr ist, dann ist auch A_2 wahr	$x_1 \leq x_2$
A_1 ist genau dann wahr, wenn A_2 wahr ist	$x_1 = x_2$
A_1 ist genau dann wahr, wenn A_2 nicht wahr ist	$x_1 = 1 - x_2$

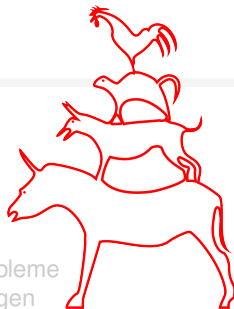
Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme
Packen, Zerlegen und Überdecken von Mengen
Traveling-Salesman-Problem



Operations Research

J. Pannek

Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

25 Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

Zusammenfassung & Ausblick

Erweiterung: Berücksichtige zwei oder mehr binären Variablen
Folgerung: Es können nicht nur logische Beziehungen zwischen mehreren binären Variablen modelliert werden.

Beispiel

Falls eine neue Smartphone-Variante zusätzlich produziert werden soll, dann muss die Fabrik erweitert werden. In der erweiterten Fabrik können maximal 300 Smartphones pro Tag zusätzlich produziert werden.

Zusätzliche Produktionsmenge: $x \in \mathbb{R}^+$ mit $x \leq 300$.

Indikatorvariable für Fabrikerweiterung:

$$y = \begin{cases} 1, & x > 0, \text{ d.h. Fabrik ist zu erweitern,} \\ 0 & \text{andernfalls, d.h. } x = 0. \end{cases}$$

$$x - 300y \leq 0$$

Fortsetzung Beispiel

Der Deckungsbeitrag pro Smartphone sei 25 GE, die Kosten für die Fabrikerweiterung betragen 2.000 GE (Abschreibung pro Tag).

Berücksichtigung der Fixkosten in der Zielfunktion:

$$\max \quad 25x - 2.000y \quad (1)$$

$$x - 300y \leq 0 \quad (2)$$

$$x \leq 300 \quad (3)$$

$$x \geq 0 \quad (4)$$

$$y \in \{0, 1\} \quad (5)$$

Frage: Welche Werte außer 300 kann der Koeffizient von y in (2) annehmen, damit der gewünschte Effekt eintritt?

Um solche Beziehungen unabhängig von konkreten Parameterwerten schreiben zu können verwendet man die **Big-M Notation**.

$$\begin{aligned} \max \quad & 25x - 2.000y \\ & x - M \cdot y \leq 0 \\ & x \geq 0 \\ & x \leq 300 \\ & y \in \{0, 1\} \end{aligned}$$

Big-M

M (sprich „Big M“) ist ein konstanter Koeffizient, der eine bekannte **obere Schranke** für x darstellt, d.h. $x \in [0, M]$.

Eigenschaft

M ist eine **hinreichend** große Zahl.

Operations Research

J. Pannek

Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

25 Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

Zusammenfassung & Ausblick

Allgemein soll modelliert werden:

Grundproblem

Aus $x > 0$ folgt $y = 1$ ($y \in \{0, 1\}, x \in \mathbb{R}^+$).

Weitere Werte von Modellparametern sind nicht bekannt.

Modellierung mit Big-M

Führe einen **hinreichend großen** konstanten Parameter M (Big-M) ein.

$$x \leq M \cdot y$$

Falls $y = 0$ dann muss $x = 0$, falls $y = 1$ darf x zwischen 0 und M liegen.

Der tatsächliche Wert von M hängt von den konkreten Werten der übrigen Modellparameter ab. Solange M hinreichend groß ist, spielt der tatsächliche Wert für das Modell keine Rolle.

Grundproblem

Aus $x > 0$ folgt $y = 1$ ($y \in \{0, 1\}, x \geq 0$).

Alternative, aber **ungenügende** Modellierungsideen:

a) $x \leq y$

Positiv: nur bei $y = 1$ kann $x > 0$ gelten.

Negativ: x kann keine Werte größer 1 annehmen.

b) $(1 - y) \cdot x = 0$

Positiv: bei $y = 1$ sind für x beliebige Werte möglich.

Negativ: Multiplikation von Entscheidungsvariablen \rightarrow **nichtlineares** Modell.

Gegebenes Modell mit Mengenbeziehung „wenn $x > 0$, dann $y = 1$ “

$$0 \leq 2x \leq 20 \quad (1)$$

$$x \leq M \cdot y \quad (2)$$

$$y \in \{0, 1\} \quad (3)$$

Wie groß muss M mindestens gewählt werden?

Aus Restriktion (1) folgt, dass zulässige Werte von x in $[0, 10]$ liegen.

Restriktion (2) darf dieses Intervall nicht einschränken. Korrekt sind demnach **alle Werte für $M \geq 10$** .

Einschränkung des Lösungsraums

Zur Lösung des Modells sollte M so groß wie nötig, aber so klein wie möglich gewählt werden.

Problem

Eine Variable x soll *entweder* den Wert 0 haben *oder* zwischen einer unteren Schranke m und einer oberen Schranke M liegen.

Anwendung Big-M Verfahren, aber für **obere und untere** Schranke.

MIP-Modell

$$y = \begin{cases} 1, & \text{falls } x \in [m, M] \\ 0, & \text{falls } y = 0 \end{cases}$$

$$x \leq M \cdot y \quad y = 0 \text{ erzwingt } x = 0, \text{ sonst } x \leq M$$

$$x \geq m \cdot y \quad y = 0 \text{ erzwingt } x = 0, \text{ sonst } x \geq m$$

Bei vorherigen Beispielen galt stets $m = 0$.

Beispiel

In welchem Umfang soll ein landwirtschaftlicher Betrieb seine 100 Hektar Land für

- ▶ die Rinderzucht,
- ▶ den Getreideanbau und
- ▶ den Anbau von Biogemüse

nutzen, so dass er seinen Gewinn maximiert?

Restriktionen und Daten:

- ▶ Rinderzucht erfordert einen Stall zu Kosten von 200 GE/Periode.
- ▶ Getreideanbau erfordert den Kauf von Maschinen für 100 GE/Periode
- ▶ Rinderzucht und Biogemüseanbau können nicht gleichzeitig erfolgen.
- ▶ Mangels Personal können höchstens 20 Hektar für den Biogemüseanbau verwendet werden.
- ▶ Für jeden Hektar, der für die Viehzucht genutzt wird, fällt ein Deckungsbeitrag von 17 GE an (Getreide 14 GE, Biogemüse 23 GE)

Operations Research

J. Pannek

Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

25 Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

Zusammenfassung & Ausblick

Kontinuierliche Entscheidungsvariablen:

- ▶ x_R für Rinderzucht verwendete Hektar
- ▶ x_G für Getreideanbau verwendete Hektar
- ▶ x_B für Biogemüseanbau verwendete Hektar

Binäre Entscheidungsvariablen:

- ▶ $y_R = 1$ Rinderzucht wird betrieben, $y_R = 0$ sonst.
- ▶ $y_G = 1$ Getreide wird angebaut, $y_G = 0$ sonst.
- ▶ $y_B = 1$ Biogemüse wird angebaut, $y_B = 0$ sonst.

Zielfunktion

$$\max z = 17x_R + 14x_G + 23x_B - 200y_R - 100y_G$$

Restriktionen

$$x_R + x_G + x_B \leq 100$$

Gesamtfläche

$$x_R \leq M_R \cdot y_R$$

$$x_G \leq M_G \cdot y_G$$

$$x_B \leq 20 \cdot y_B$$

$$y_R + y_B \leq 1$$

Rinder und Biog. nicht gleichzeitig

$$x_i \geq 0$$

$i = R, G, B$

Sinnvolle Werte für M_R , M_G und M_B :

$M_R = M_G = 100$ wegen Beschränkung Gesamtfläche

Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme
Packen, Zerlegen und Überdecken von Mengen
Traveling-Salesman-Problem



Operations Research

J. Pannek

Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

260 Ausgewählte kombinatorische Optimierungsprobleme

Packen, Zerlegen und Überdecken von Mengen

Traveling-Salesman-Problem

Zusammenfassung & Ausblick

▶ **Zuordnungsprobleme**

lineares und quadratisches Zuordnungsproblem, Stundenplanungsprobleme

▶ **Reihenfolgeprobleme**

Traveling Salesman Problem, allg. Tourenplanung, Maschinenbelegungsprobleme

▶ **Gruppierungsprobleme**

Zuschnitt- und Packprobleme, Fließbandabstimmung, Losgrößenplanung, Clusteranalyse

▶ **Auswahlprobleme**

z.B. Rucksack-, Set-Partitioning- und Set-Covering-Problem

Mengenüberdeckungsproblem – Set-Covering-Problem

Beispiel Allokation von Rettungswagen

Beispiel

Wo sollen Rettungswagen positioniert werden, sodass sämtliche Ortsteile von Bremen innerhalb von 5 Minuten von mindestens einem Rettungswagen erreicht werden können?

Ortsteil	Stellplatz							...
	1	2	3	4	5	6	7	
Bürgerpark	1					1		
Neustadt							1	
Ostertor							1	
Horn		1		1				
Lehe		1	1					
Lehesterdeich			1					
Schwachhausen	1			1		1		
Äberseehafen					1		1	
...								



Operations Research

J. Pannek

Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

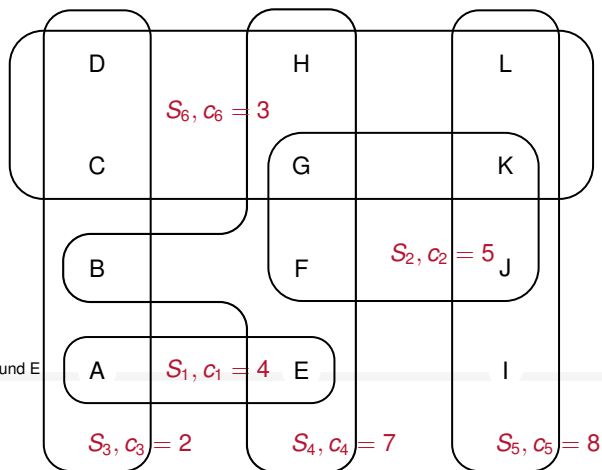
Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

26 Packen, Zerlegen und Überdecken von Mengen

Traveling-Salesman-Problem

Zusammenfassung & Ausblick



Menge $U = \{A, B, C, D, E, F, G, H, I, J, K, L\}$

Menge $S = \{S_1, S_2, S_3, S_4, S_5, S_6\}$, mit z.B. $S_2 = \{G, K, F, J\}$ und $c_2 = 5$

Mengenüberdeckungsproblem (Set-Covering-Problem)

Gegeben sind eine Menge U von m Objekten sowie n mit Kosten c_j bewertete Teilmengen S_j von U . Gesucht ist eine **kostenminimale Auswahl von Teilmengen S_j** , sodass jedes der m Objekte in **mindestens einer** der ausgewählten Teilmengen enthalten ist.

Parameter: $a_{ij} = 1$, falls Objekt $i \in U$ in Teilmenge S_j enthalten ist, 0 sonst.

$$\begin{aligned} \min \quad & \sum_{j=1}^n c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^n a_{ij} x_j \geq 1 \quad \text{für } i = 1, \dots, m \\ & x_1, \dots, x_n \in \{0, 1\} \end{aligned}$$

(Gewichtetes) Mengenüberdeckungsproblem (Set-Covering-Problem)
Minimierung und nur \geq -Restriktionen; rechte Seite nur 1; übrige
Matrixkoeffizienten nur 0 oder 1

Mengenzerlegungsproblem (Set-Partitioning-Problem)

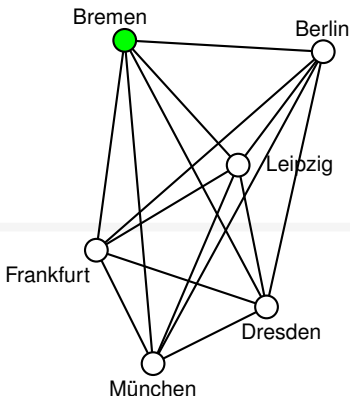
Gegeben sind eine Menge U von m Objekten sowie n mit *Kosten* c_j bewertete Teilmengen S_j von U . Gesucht ist eine **kostenminimale Auswahl von Teilmengen** S_j , sodass jedes der m Objekte in **genau einer** der ausgewählten Teilmengen enthalten ist.

Mengenpackproblem (Set-Packing-Problem)

Gegeben sind eine Menge U von m Objekten sowie n mit *Nutzen* u_j bewertete Teilmengen S_j von U . Gesucht ist eine **kostenminimale Auswahl von Teilmengen** S_j , sodass jedes der m Objekte in **höchstens einer** der ausgewählten Teilmengen enthalten ist.

Traveling-Salesman-Problem (TSP)

Welches ist die kürzeste Tour, so dass der Handelsreisende jeden Ort genau einmal besucht und schließlich am Startort wieder ankommt?



Gegeben:

- ▶ $V = \{1 \dots n\}$ Menge von Knoten, z.B. Kundenorten
- ▶ $G = (V, E, c)$ vollständiger Graph
- ▶ bewertete Kanten, z.B. c_{ij} Entfernung zwischen dem Knoten i und dem Knoten j
- ▶ symmetrisches TSP: $c_{ij} = c_{ji}$
- ▶ asymmetrisches TSP: $c_{ij} \neq c_{ji}$

Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

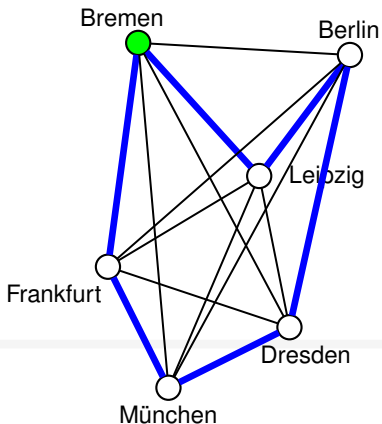
Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

Packen, Zerlegen und Überdecken von Mengen

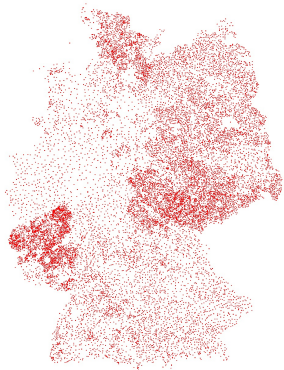
269 Traveling-Salesman-Problem

Zusammenfassung & Ausblick



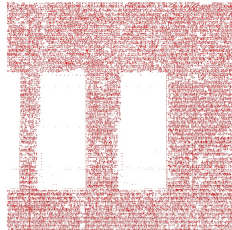
- ▶ Lagerhaus – Routing eines Pickers zum Einsammeln der Ware
- ▶ Genetik – DNS-Sequenzierung
- ▶ Raumfahrt – Treibstoffeinsparung bei der Repositionierung von Satelliten
- ▶ Halbleiterindustrie – Design von Schaltkreisen auf Chips
- ▶ Telekommunikation – Einsammeln von Münzen aus Münztelefonen
- ▶ Telekommunikation – Routing in ringförmigen Glasfasernetzen
- ▶ Energie – Design von Stromnetzen, insb. letzte Meile
- ▶ Teilprobleme komplexerer Tourenplanungsprobleme

Deutschlandtour



15.112 Städte, Tourlänge ca. 66.000km,
Optimale Lösung gefunden am 20.4.2001,
110 CPUs, 22 CPU-Jahre Rechenzeit

Integrierte Schaltkreise



85.900 Knoten, Optimale Lösung
gefunden 2006,
136 CPU-Jahre Rechenzeit

Geschichte, Anwendungen, Google-Maps
TSP-Planer, iPhone App für TSP:
<http://www.tsp.gatech.edu/apps/>

Operations Research

J. Pannek

Wiederholung

Ganzzahlige Entscheidungsvariablen

Logische Verknüpfung von Restriktionen

Mengenmäßige Verknüpfung von Restriktionen

Ausgewählte kombinatorische Optimierungsprobleme

Packen, Zerlegen und Überdecken von Mengen

269 Traveling-Salesman-Problem

Zusammenfassung & Ausblick

$x_{ij} = 1$, wenn in der Tour Knoten j unmittelbar auf Knoten i folgt, 0 sonst.

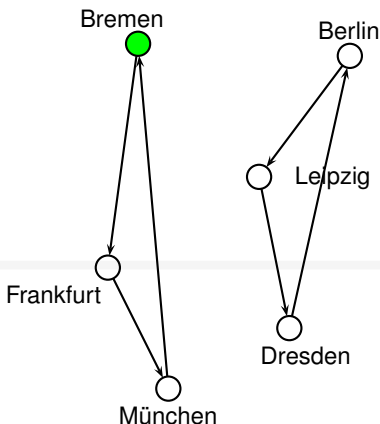
$$\sum_{j=1}^n x_{ij} = 1, \quad \forall i \in V$$

$$\sum_{j=1}^n x_{ji} = 1, \quad \forall i \in V$$

Subtour Eliminierung:

In einer zulässigen Tour dürfen für jede echte Teilmenge S der Menge aller Knoten V höchstens $|S| - 1$ Kanten verwendet werden.

Unerwünschte Subtour



Entscheidungsvariablen: $x_{ij} = 1$, wenn j auf i in der Tour folgt, 0 sonst.

$$\min \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1, j \neq i}^n x_{ij} = 1 \quad \text{für } i = 1, \dots, n$$

$$\sum_{j=1, j \neq i}^n x_{ji} = 1 \quad \text{für } i = 1, \dots, n$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \quad \text{für } S \subset \{1, \dots, n\} \text{ und } 2 \leq |S| \leq \frac{n}{2}$$

$$x_{ij} \in \{0, 1\} \quad \text{für } i, j = 1, \dots, n$$

Bei $n = 50$ Knoten mehr als 33 Millionen Restriktionen zur Subtour-Eliminierung.

Fragen zur Wiederholung:

- ▶ Was bedeutet die Begriffe logische Abhängigkeiten und Mengenbeziehungen mit Big-M?
- ▶ Was bezeichnen die kombinatorischen Optimierungsprobleme Knapsack-, Set-Covering-, Set-Partitioning-, Set-Packing-, Traveling-Salesman-Problem?

Klausurrelevante Literatur

Modellierung **ganzzahliger** Optimierungsprobleme Domschke/Drexler Kapitel 6.1, 6.2, 6.5 oder Nickel/Stein/Waldman Kapitel 5.1, 5.2 oder Suhl/Mellouli Kapitel 4.1–4.4, 4.7

Ausblick

- ▶ Tutorium: Modellierung **ganzzahliger** Optimierungsprobleme
- ▶ Vorlesung: Lösung von IP/MIP mit **Branch & Bound**, siehe Domschke/Drexler Kapitel 6.4, 6.5 oder Nickel/Stein/Waldman Kapitel 5.4.1 oder Suhl/Mellouli Kapitel 5.3

Wiederholung

Vollständige
Enumeration

Branching –
Verzweigen in
Teilprobleme

Bounding – Verwerfen
von Teilproblemen

Ablauf
Branch-and-Bound
Verfahren

B&B für ein
Zuordnungsproblem

B&B mit
LP-Relaxierung für ein
Rucksackproblem

Zusammenfassung &
Ausblick

Teil IX

Ganzzahlige und kombinatorische Optimierung II – Branch-and-Bound

Wiederholung

Vollständige Enumeration

Branching – Verzweigen in Teilprobleme

Bounding – Verwerfen von Teilproblemen

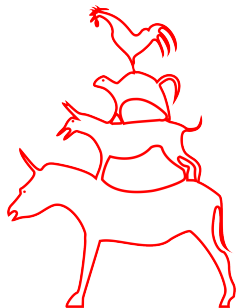
Lower Bound

Upper Bound und Relaxierung

Ablauf Branch-and-Bound Verfahren

B&B für ein Zuordnungsproblem

B&B mit LP-Relaxierung für ein Rucksackproblem



273 Wiederholung

Vollständige Enumeration

Branching – Verzweigen in Teilprobleme

Bounding – Verwerfen von Teilproblemen

Ablauf Branch-and-Bound Verfahren

B&B für ein Zuordnungsproblem

B&B mit LP-Relaxierung für ein Rucksackproblem

Zusammenfassung & Ausblick

Vollständige
EnumerationBranching –
Verzweigen in
TeilproblemeBounding – Verwerfen
von TeilproblemenAblauf
Branch-and-Bound
VerfahrenB&B für ein
ZuordnungsproblemB&B mit
LP-Relaxierung für ein
RucksackproblemZusammenfassung &
Ausblick

Die Entscheidungsvariable x_i wird auch als **binäre Indikatorvariable** bezeichnet.

$$x_i = \begin{cases} 1, & \text{falls Aussage } A_i \text{ wahr ist} \\ 0, & \text{andernfalls.} \end{cases} \quad i = 1, 2$$

Aussage	Logische Verknüpfung
A_1 und/oder A_2 ist wahr	$x_1 + x_2 \geq 1$
A_1 und A_2 sind wahr	$x_1 + x_2 = 2$
Weder A_1 noch A_2 sind wahr	$x_1 + x_2 = 0$
Wenn A_1 wahr ist, dann ist auch A_2 wahr	$x_1 \leq x_2$
A_1 ist genau dann wahr, wenn A_2 wahr ist	$x_1 = x_2$
A_1 ist genau dann wahr, wenn A_2 nicht wahr ist	$x_1 = 1 - x_2$

Allgemein soll modelliert werden:

Grundproblem

Aus $x > 0$ folgt $y = 1$ ($y \in \{0, 1\}, x \in \mathbb{R}^+$).

Weitere Werte von Modellparametern sind nicht bekannt.

Modellierung mit Big-M

Führe einen **hinreichend großen** konstanten Parameter M (Big-M) ein.

$$x \leq M \cdot y$$

Falls $y = 0$ dann muss $x = 0$, falls $y = 1$ darf x zwischen 0 und M liegen.

Der tatsächliche Wert von M hängt von den konkreten Werten der übrigen Modellparameter ab. Solange M hinreichend groß ist, spielt der tatsächliche Wert für das Modell keine Rolle.

275 Wiederholung

Vollständige Enumeration

Branching – Verzweigen in Teilprobleme

Bounding – Verwerfen von Teilproblemen

Ablauf
Branch-and-Bound Verfahren

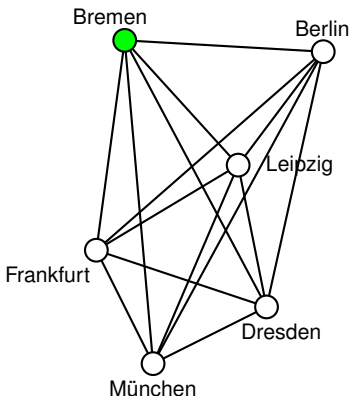
B&B für ein Zuordnungsproblem

B&B mit LP-Relaxierung für ein Rucksackproblem

Zusammenfassung & Ausblick

Traveling-Salesman-Problem (TSP)

Welches ist die kürzeste Tour, so dass der Handelsreisende jeden Ort genau einmal besucht und schließlich am Startort wieder ankommt?



Gegeben:

- ▶ $V = \{1 \dots n\}$ Menge von Knoten, z.B. Kundenorten
- ▶ $G = (V, E, c)$ vollständiger Graph
- ▶ bewertete Kanten, z.B. c_{ij} Entfernung zwischen dem Knoten i und dem Knoten j
- ▶ symmetrisches TSP: $c_{ij} = c_{ji}$
- ▶ asymmetrisches TSP: $c_{ij} \neq c_{ji}$

276 Wiederholung

Vollständige Enumeration

Branching – Verzweigen in Teilprobleme

Bounding – Verwerfen von Teilproblemen

Ablauf Branch-and-Bound Verfahren

B&B für ein Zuordnungsproblem

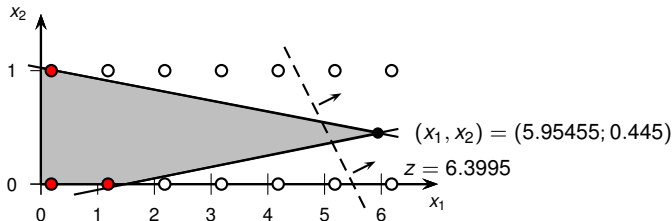
B&B mit LP-Relaxierung für ein Rucksackproblem

Zusammenfassung & Ausblick

Beispiel ganzzahliges Modell:

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{s.t.} \quad & 0.011x_1 + x_2 \leq 1.01 \\ & -0.011x_1 + x_2 \leq -0.021 \\ & x_1, x_2 \in \mathbb{N}_0 \end{aligned}$$

Wie lautet eine optimale Lösung?



Problem

Ganzzahlige Probleme können im Allgemeinen nicht mit dem Simplex-Algorithmus gelöst werden.

27 Wiederholung

Vollständige Enumeration

Branching – Verzweigen in Teilprobleme

Bounding – Verwerfen von Teilproblemen

Ablauf Branch-and-Bound Verfahren

B&B für ein Zuordnungsproblem

B&B mit LP-Relaxierung für ein Rucksackproblem

Zusammenfassung & Ausblick

Wiederholung

Wiederholung

Vollständige Enumeration

27x Vollständige Enumeration

Branching – Verzweigen in Teilprobleme

Branching – Verzweigen in Teilprobleme

Bounding – Verwerfen von Teilproblemen

Bounding – Verwerfen von Teilproblemen

Lower Bound

Upper Bound und Relaxierung

Ablauf Branch-and-Bound Verfahren

Ablauf Branch-and-Bound Verfahren

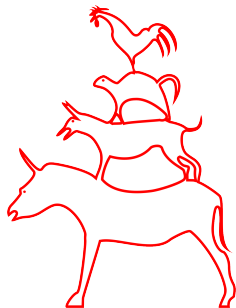
B&B für ein Zuordnungsproblem

B&B für ein Zuordnungsproblem

B&B mit LP-Relaxierung für ein Rucksackproblem

B&B mit LP-Relaxierung für ein Rucksackproblem

Zusammenfassung & Ausblick



Rucksackproblem

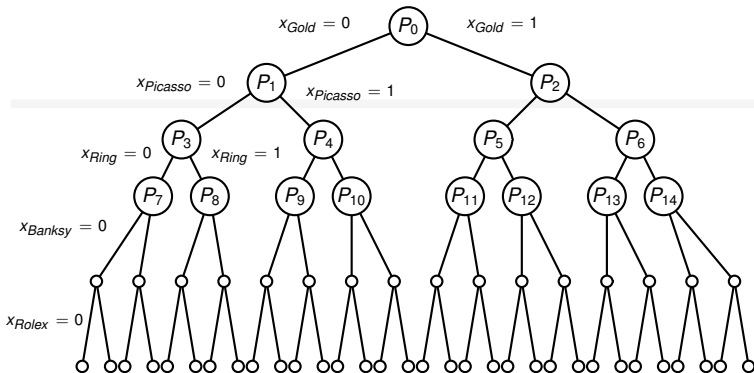
Aus n Objekten mit Gewichten $w_i \geq 0$ ($i = 1, \dots, n$) und Nutzenwerten $u_i \geq 0$ ($i = 1, \dots, n$) soll eine nutzenmaximale Auswahl getroffen werden, sodass ein maximales Gesamtgewicht C nicht überschritten wird.

Ein Dieb auf Beutezug – Beispiel für Problem Instanz

$$\begin{aligned} \max \quad & 10x_{Gold} + 9x_{Picasso} + 12x_{Ring} + 5x_{Banksy} + 9x_{Rolex} \\ \text{s.t.} \quad & 5x_{Gold} + 6x_{Picasso} + 12x_{Ring} + 10x_{Banksy} + 12x_{Rolex} \leq 25 \\ & x_j \in \{0, 1\} \quad \forall j \end{aligned}$$

Vollständige Enumeration für das Rucksackproblem

Für jeden der $n = 5$ Gegenstände gibt es zwei Zustände.



$\Rightarrow 2^5 = 32$ Lösungen (Blattknoten)

Wiederholung

279 **Vollständige Enumeration**

Branching – Verzweigen in Teilprobleme

Bounding – Verwerfen von Teilproblemen

Ablauf
Branch-and-Bound Verfahren

B&B für ein Zuordnungsproblem

B&B mit LP-Relaxierung für ein Rucksackproblem

Zusammenfassung & Ausblick

Rucksack-Problem mit n Objekten: 2^n Lösungen

Traveling-Salesman-Problem mit n Städten: $(n - 1)!$ Lösungen

n	$\log n$	n^2	2^n	$n!$
10	3.32	100	$1.02 \cdot 10^3$	$3.6 \cdot 10^6$
100	6.64	10.000	$1.27 \cdot 10^{30}$	$9.33 \cdot 10^{157}$
1000	9.97	1.000.000	$1.07 \cdot 10^{300}$	$4.02 \cdot 10^{2567}$

Falls ein PC 10^6 TSP-Lösungen pro Sekunden berechnet, dauert die Lösung mittels vollständige Enumeration bei

23 Knoten 0.82 Milliarden Jahre und bei

24 Knoten 19.67 Milliarden Jahre.

Alter des Universums: ≈ 13.8 Milliarden Jahre

Wiederholung

Wiederholung

Vollständige Enumeration

Vollständige
Enumeration

Branching – Verzweigen in Teilprobleme

280 **Branching –
Verzweigen in
Teilprobleme**

Bounding – Verwerfen von Teilproblemen

Bounding – Verwerfen
von Teilproblemen

Lower Bound

Upper Bound und Relaxierung

Ablauf
Branch-and-Bound
Verfahren

B&B für ein
Zuordnungsproblem

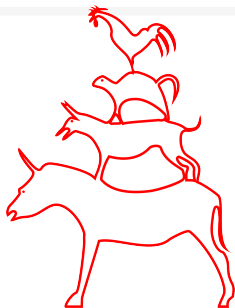
Ablauf Branch-and-Bound Verfahren

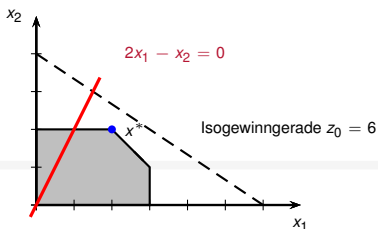
B&B mit
LP-Relaxierung für ein
Rucksackproblem

B&B für ein Zuordnungsproblem

Zusammenfassung &
Ausblick

B&B mit LP-Relaxierung für ein Rucksackproblem





$$\begin{aligned}
 P_0 : \quad & \max \quad x_1 + 1.5x_2 \\
 \text{s.t.} \quad & x_1 \leq 3 \\
 & x_2 \leq 2 \\
 & x_1 + x_2 \leq 4 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

Optimale Lösung von P_0 : $\mathbf{x}^* = (2, 2)$ mit $z^* = 5$.

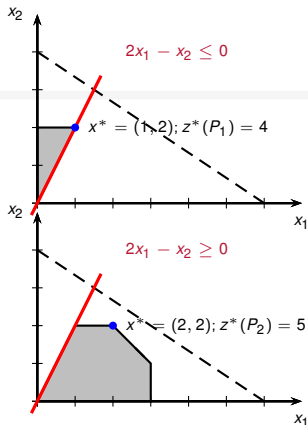
Das Problem P_0 kann durch Hinzunahme einer *zusätzlichen Restriktion*, z.B. $2x_1 - x_2 = 0$, in zwei Teilprobleme P_1, P_2 *verzweigt* (to branch) bzw. zerlegt werden:

$$P_1 : P_0 \text{ zzgl. } 2x_1 - x_2 \geq 0$$

$$P_2 : P_0 \text{ zzgl. } 2x_1 - x_2 \leq 0$$

Keine zulässige Lösung geht verloren, d.h. $\mathbb{M}(P_0) = \mathbb{M}(P_1) \cup \mathbb{M}(P_2)$.

Der optimale Wert z^* von P_0 ist $z^*(P_0) = \max\{z^*(P_1), z^*(P_2)\}$.



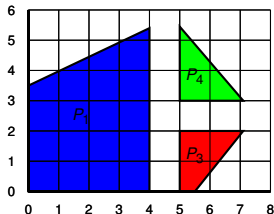
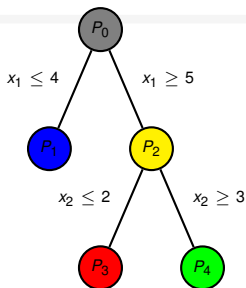
$$\begin{aligned}
 P_1 : \quad & \max && x_1 + 1.5x_2 \\
 & \text{s.t.} && x_1 \leq 3 \\
 & && x_2 \leq 2 \\
 & && x_1 + x_2 \leq 4 \\
 & && 2x_1 - x_2 \leq 0 \\
 & && x_1, x_2 \geq 0
 \end{aligned}$$

$$\begin{aligned}
 P_2 : \quad & \max && x_1 + 1.5x_2 \\
 & \text{s.t.} && x_1 \leq 3 \\
 & && x_2 \leq 2 \\
 & && x_1 + x_2 \leq 4 \\
 & && 2x_1 - x_2 \geq 0 \\
 & && x_1, x_2 \geq 0
 \end{aligned}$$

Branching

Die Menge $\mathbb{M}(P_0)$ der **zulässigen** Lösungen von P_0 wird durch Hinzunahme weiterer Restriktionen **partitioniert**.

Verzweigen bei ganzzahligen Variablenwerten, z.B. $x_1, x_2 \in \mathbb{N}_0$.



Durch Verzweigung gehen keine Informationen verloren. Anhand welcher Variablen/Werte zu verzweigen ist, hängt vom konkreten Verfahren ab. Zerlegung in mehr als zwei Teilprobleme auch möglich.

Wiederholung

Vollständige
Enumeration283 Branching –
Verzweigen in
TeilproblemeBounding – Verwerfen
von TeilproblemenAblauf
Branch-and-Bound
VerfahrenB&B für ein
ZuordnungsproblemB&B mit
LP-Relaxierung für ein
RucksackproblemZusammenfassung &
Ausblick

Wiederholung

Vollständige Enumeration

Branching – Verzweigen in Teilprobleme

Bounding – Verwerfen von Teilproblemen

Lower Bound

Upper Bound und Relaxierung

Ablauf Branch-and-Bound Verfahren

B&B für ein Zuordnungsproblem

B&B mit LP-Relaxierung für ein Rucksackproblem



Operations Research

J. Pannek

Wiederholung

Vollständige
Enumeration

Branching –
Verzweigen in
Teilprobleme

**283 Bounding – Verwerfen
von Teilproblemen**

Lower Bound

Upper Bound und
Relaxierung

Ablauf
Branch-and-Bound
Verfahren

B&B für ein
Zuordnungsproblem

B&B mit
LP-Relaxierung für ein
Rucksackproblem

Zusammenfassung &
Ausblick

Bounding

Mittels **Bounding** wird für jedes Teilproblem geprüft, ob dieses zu verzweigen ist oder nicht.

Dazu berechnet man

- ▶ *untere Schranken* (**lower bound, LB**) für den optimalen Zielfunktionswert von P_0 und
- ▶ *obere Schranken* (**upper bound, UB**) für den optimalen Zielfunktionswert eines Teilproblems P_i und
- ▶ entscheidet anhand dieser Schranken, ob ein Teilproblem P_i zu verzweigen ist oder nicht (*Auslotung*).

Sinn des Bounding

Teilprobleme identifizieren, deren zulässiger Bereich **keine Optimallösung** des Ausgangsproblems P_0 enthält

Untere Schranke (lower bound, LB)

Eine **untere Schranke** LB eines Maximierungsproblems P_0 ist ein Wert, der **stets kleiner-gleich** dem maximalen Zielfunktionswert f^* von P_0 ist, d.h. $f^* \geq LB$

Eine LB bezieht sich auf das (globale) Ausgangsproblem P_0 .

Wie berechnet man eine LB?

- Der kleinste (schlechteste) Zielfunktionswert für ein Maximierungsproblem ist stets $-\infty$, d.h. $LB = -\infty$ gilt immer.
- Der **Wert jeder zulässigen** Lösung eines Maximierungsproblems P_0 ist eine **untere Schranke**, d.h. $LB = f(x)$, falls x für P_0 zulässig ist.

Wie lautet eine *untere Schranken (LB)* für das folgende MIP?

$$\begin{array}{rcll} \max & 5x_1 & +x_2 & \\ & x_1 & & \leq 5 \\ & 2x_1 & +x_2 & \leq 7.5 \\ & x_1 & & \in \mathbb{N}_0 \\ & & x_2 & \geq 0 \end{array}$$

Wie lautet eine **obere Schranken (UB)** für das folgenden Zuordnungsproblem (**Minimierungsproblem!**)?

	Rücken	Brust	Schmetterling	Freistil
Christine	2.56	3.07	2.90	2.26
Janine	2.63	3.01	3.12	2.35
Larissa	2.71	2.95	2.96	2.29
Laura	2.60	2.86	3.08	2.41

Wiederholung

Vollständige
EnumerationBranching –
Verzweigen in
TeilproblemeBounding – Verwerfen
von Teilproblemen

28 Lower Bound

Upper Bound und
RelaxierungAblauf
Branch-and-Bound
VerfahrenB&B für ein
ZuordnungsproblemB&B mit
LP-Relaxierung für ein
RucksackproblemZusammenfassung &
Ausblick

Obere Schranke (upper bound, UB)

Eine **obere Schranke** $UB(P_i)$ eines Maximierungsproblems P_i ist ein Wert, der **stets größer-gleich** dem maximalen Zielfunktionswert $f^*(P_i)$ von P_i ist, d.h. $f^*(P_i) \leq UB(P_i)$

UB werden für (lokale) Teilprobleme P_i berechnet.

Wie berechnet man einen UB für ein Teilproblem P_i ?

Vereinfache das Teilproblem P_i , z.B. durch Weglassen von Restriktionen, und lösen das so **relaxierte** Problem P_i^R .

Idee Relaxierung

Ersetze ein **schwierig** zu lösendes Optimierungsproblem P durch ein leichter zu lösendes Optimierungsproblem P^R , dessen optimaler Wert aber **mindestens so groß** wie der von P ist: $z^*(P) \leq z^*(P^R)$.

Wie relaxiert man ein Optimierungsproblem?

1. Vergrößerung des Lösungsraums, sodass $\mathbb{M}(P) \subseteq \mathbb{M}(P^R)$ oder
2. Ersetzen der Zielfunktion $f(x)$ durch $f^R(x)$, sodass $f^R(x) \geq f(x)$ für alle $x \in \mathbb{M}(P)$ (Lagrange Relaxierung, **hier nicht weiter betrachtet**)

$\mathbb{M}(P)$ steht für die Menge der zulässigen Lösung von Problem P

LP-Relaxierung

Entscheidungsvariablen mit **ganzzahligem Wertebereich** werden durch Variablen mit einem **kontinuierlichen Wertebereich** ersetzt.

Ausgangsproblem P_0 :

$$\begin{aligned} P_0 : \quad \max f(x) &= 4x_1 - x_2 \\ 7x_1 - 2x_2 &\leq 14 \\ x_2 &\leq 3 \\ 2x_1 - 2x_2 &\leq 3 \\ x_1, x_2 &\in \mathbb{Z}_+ \end{aligned}$$

LP-Relaxierung von P_0 :

$$\begin{aligned} P_0^R : \quad \max f(x) &= 4x_1 - x_2 \\ 7x_1 - 2x_2 &\leq 14 \\ x_2 &\leq 3 \\ 2x_1 - 2x_2 &\leq 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Lösung der **LP-Relaxierung** P_0^R mittels Simplex: $x^* = (\frac{20}{7}, 3)$,
 $f(x^*) = \frac{59}{7} \sim 8.42$.

Folgerung

Der Wert einer optimalen Lösung von P_0 kann nicht größer als 8.42 sein, d.h. $UB(P_0) = 8.42$; wg. Ganzzahligkeit gilt sogar $UB(P_0) = 8$

Operations Research

J. Pannek

Wiederholung

Vollständige
Enumeration

Branching –
Verzweigen in
Teilprobleme

Bounding – Verwerfen
von Teilproblemen

Lower Bound
Upper Bound und
Relaxierung

Ablauf
Branch-and-Bound
Verfahren

B&B für ein
Zuordnungsproblem

B&B mit
LP-Relaxierung für ein
Rucksackproblem

Zusammenfassung &
Ausblick

Kombinatorischen Relaxierung

Ignoriere für ein Problem P_0 ein oder mehrere Restriktionen.

Das relaxierte Problem P_0^R muss einen mindestens gleich guten Zielfunktionswert wie P_0 haben, da das Weglassen von Restriktionen nur zu einer Verbesserung führen kann.

Beispiele:

- ▶ *Traveling-Salesman-Problem* – Verzicht auf Restriktionen zur Vermeidung von Subtouren
- ▶ *Lineares Zuordnungsproblem* für Schwimmteam (VL07), ordne jeder Schwimmerin genau eine Disziplin zu. **Relaxierung:** ordne eine Disziplin keiner, einer oder mehr als einer Schwimmerin zu.

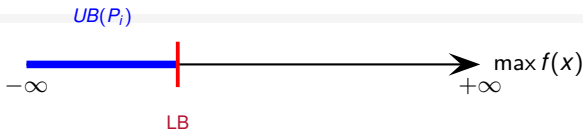
Muss ein Teilproblem verzweigt werden? 1/3 – Fall a

Ein Teilproblem P_i muss **nicht mehr verzweigt** werden, wenn ...

Fall a: $UB(P_i) \leq LB$

... die obere Schranke $UB(P_i)$ des Teilproblems P_i kleiner ist als die beste untere Schranke LB des Ausgangsproblems P_0 .

Wir kennen bereits eine Lösung, die besser als die bestmögliche Lösung von P_i ist. P_i muss nicht verzweigt werden und heißt daher **ausgelotet**.



Notation: P bzw. P_0 – originäres Problem

P_i – durch Verzweigung entstandenes Teilproblem i , P_i^R – Relaxierung von P_i

LB – untere Schranke von P_0 , $UB_i = UB(P_i)$ – obere Schranke von Teilproblem P_i

Wiederholung

Vollständige
EnumerationBranching –
Verzweigen in
TeilproblemeBounding – Verwerfen
von Teilproblemen

Lower Bound

29 Upper Bound und
RelaxierungAblauf
Branch-and-Bound
VerfahrenB&B für ein
ZuordnungsproblemB&B mit
LP-Relaxierung für ein
RucksackproblemZusammenfassung &
Ausblick

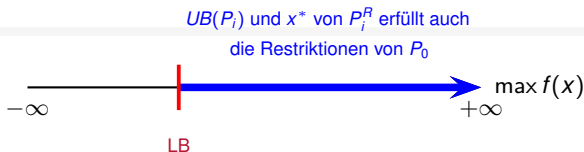
Muss ein Teilproblem verzweigt werden? 2/3 – Fall b

Ein Teilproblem P_i muss **nicht mehr verzweigt** werden, wenn ...

Fall b: $UB(P_i) > LB$ und Zulässigkeit für P_0

... die obere Schranke $UB(P_i)$ des Teilproblems P_i größer als der beste bekannte LB ist **und** die Lösung x^* für P_i^R auch für P_i (und damit auch für P_0) zulässig ist.

Wir haben eine bessere Lösung für P_0 gefunden, der LB ist zu aktualisieren, $LB := UB(P_i)$. P_i ist **ausgelotet**.



Notation: P bzw. P_0 – originäres Problem

P_i – durch Verzweigung entstandenes Teilproblem i , P_i^R – Relaxierung von P_i

LB – untere Schranke von P_0 , $UB_i = UB(P_i)$ – obere Schranke von Teilproblem P_i

Wiederholung

Vollständige
EnumerationBranching –
Verzweigen in
TeilproblemeBounding – Verwerfen
von TeilproblemenLower Bound
Upper Bound und
RelaxierungAblauf
Branch-and-Bound
VerfahrenB&B für ein
ZuordnungsproblemB&B mit
LP-Relaxierung für ein
RucksackproblemZusammenfassung &
Ausblick

Muss ein Teilproblem verzweigt werden? 3/3 – Fall c

Ein Teilproblem P_i muss **nicht mehr verzweigt** werden, wenn ...

Fall c: $M(P_i^R) = \emptyset$

... das relaxierte Problem P_i^R keine zulässige Lösung hat.

Existiert für das weniger strikte Restriktionssystem von P_i^R keine Lösung, gibt es auch keine für das strikere P_i .

Keine zulässige Lösung für $P_i^R \Rightarrow P_i$ ist nicht lösbar.



Verzweigen

Die Fälle a, b, c schließen einander aus. **Falls keiner der drei Fälle vorliegt**, ist das Teilproblem P_i zu **verzweigen**.

Problem P_i heißt **ausgelotet** und muss nicht verzweigt werden, falls:

- a. $UB(P_i) \leq LB(P_0)$ – Ausloten wg. **Bound**

Der optimale Wert von P_i ist schlechter als der beste bislang gefundene Wert

- b. $UB(P_i) > LB(P_0)$ und $x^*(P_i^R) \in \mathbb{M}(P_0)$ – Ausloten wg. **Optimalität**

Das Teilproblem P_i wurde optimal gelöst, zugleich ist die gefundene Lösung besser als die beste bislang bekannte Lösung.

- c. $\mathbb{M}(P_i^R) = \emptyset$ – Ausloten wg. **Unzulässigkeit**

Für das relaxierte Problem P_i^R gibt es keine zulässige Lösung, daher gibt es auch für P_i keine.

Wiederholung

Wiederholung

Vollständige Enumeration

Vollständige
Enumeration

Branching – Verzweigen in Teilprobleme

Branching –
Verzweigen in
Teilprobleme

Bounding – Verwerfen von Teilproblemen

Bounding – Verwerfen
von Teilproblemen

Lower Bound

Upper Bound und Relaxierung

294 Ablauf
Branch-and-Bound
Verfahren

B&B für ein
Zuordnungsproblem

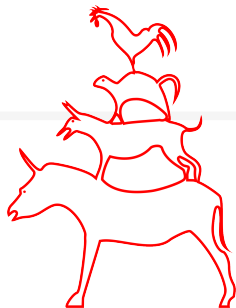
B&B mit
LP-Relaxierung für ein
Rucksackproblem

Zusammenfassung &
Ausblick

Ablauf Branch-and-Bound Verfahren

B&B für ein Zuordnungsproblem

B&B mit LP-Relaxierung für ein Rucksackproblem



Branch & Bound Verfahren

Input : Ganzzahliges Problem P_0

- 1 Bestimme LB anhand initialer Lösung (optional)
- 2 Löse die Relaxierung P_0^R von P_0
- 3 **if** P_0 ist ausgelotet **then**
 - 4 $L := \emptyset$ // L ist eine Liste mit aktiven Teilproblemen
- 5 **else**
 - 6 $L := \{P_0\}$
- 7 **while** $L \neq \emptyset$ **do**
 - 8 Wähle ein aktives Problem P_k aus L , setze $L := L \setminus \{P_k\}$.
 - 9 Löse die Relaxierung P_k^R von P_k
 - 10 **if** P_k ist ausgelotet **then**
 - 11 $L := L \setminus \{P_k\}$
 - 12 **else**
 - 13 Verzweige P_k und füge die entstehenden Teilprobleme zu L hinzu.

Output: Optimale Lösung x^* und optimaler Wert LB von P_0

Wiederholung

Vollständige
EnumerationBranching –
Verzweigen in
TeilproblemeBounding – Verwerfen
von Teilproblemen295 **Ablauf**
Branch-and-Bound
VerfahrenB&B für ein
ZuordnungsproblemB&B mit
LP-Relaxierung für ein
RucksackproblemZusammenfassung &
Ausblick

Welches Teilproblem soll als nächstes verzweigt werden?

- ▶ Last-in, first-out (**LIFO-Regel**): das zuletzt der Liste L hinzugefügte Problem wird verzweigt.
- ▶ Maximum-upper-bound (**MUB-Regel**): das aktive Problem P_i mit dem größten $UB(P_i)$ wird verzweigt.

Wie soll ein Problem verzweigt werden? Bei LP-Relaxierung:

- ▶ Verzweige beliebige nicht-ganzzahlige Variable x_i^R , mit $x_i \leq \lfloor x_i^R \rfloor$ bzw. $x_i \geq \lceil x_i^R \rceil$
- ▶ Verzweige Variable mit kleinsten nicht-ganzzahligen Anteil

Beispiel: Variable mit kleinstem nicht-ganzzahligen Anteil

Ein LP-relaxierte Lösung eines rein ganzzahligen Problems P_0 sei $x^R = (7, 0, 0.5, 4, 3.2, 4, 0.8)$. Verzweigt wird anhand von $x_5 = 3.2$, mit 0.2. hat sie den kleinsten nicht-ganzzahligen Anteil. Verzweige in zwei Probleme mit zusätzlichen Restriktionen $x_2 \leq 3$ bzw. $x_2 \geq 4$.

Wiederholung

Wiederholung

Vollständige Enumeration

Vollständige
Enumeration

Branching – Verzweigen in Teilprobleme

Branching –
Verzweigen in
Teilprobleme

Bounding – Verwerfen von Teilproblemen

Bounding – Verwerfen
von Teilproblemen

Lower Bound

Upper Bound und Relaxierung

Ablauf
Branch-and-Bound
Verfahren

Ablauf Branch-and-Bound Verfahren

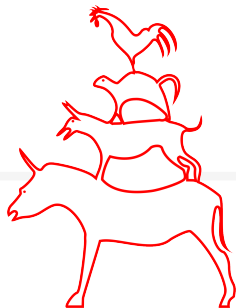
B&B für ein Zuordnungsproblem

29 **B&B für ein
Zuordnungsproblem**

B&B mit
LP-Relaxierung für ein
Rucksackproblem

B&B mit LP-Relaxierung für ein Rucksackproblem

Zusammenfassung &
Ausblick



Es gibt vier Aufträge A1 bis A4 sowie vier Maschinen M1 bis M4.

- ▶ Jede Maschine muss genau einen Auftrag bearbeiten.
- ▶ Jeder Auftrag muss auf genau einer Maschine bearbeitet werden.
(Relaxierung!)

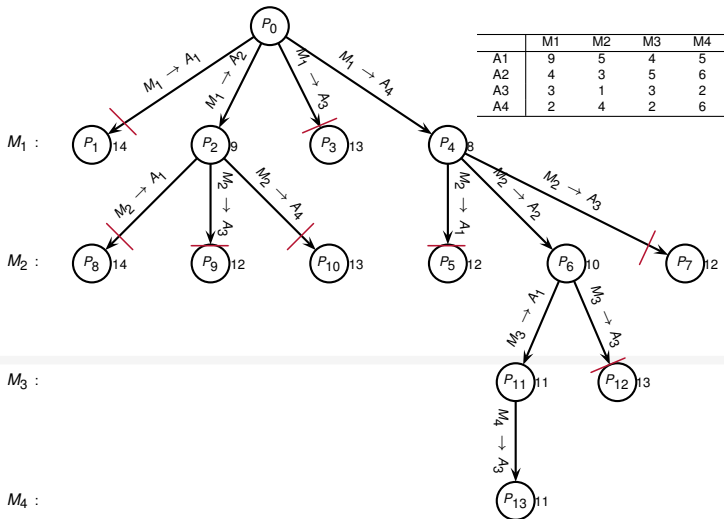
Bearbeitungsdauern in Stunden:

	M1	M2	M3	M4
A1	9	5	4	5
A2	4	3	5	6
A3	3	1	3	2
A4	2	4	2	6

Problem

Wie müssen die Maschinen mit Aufträgen belegt werden, damit die gesamte Produktionszeit minimal ist?

B&B für ein Zuordnungsproblem



	M1	M2	M3	M4
A1	9	5	4	5
A2	4	3	5	6
A3	3	1	3	2
A4	2	4	2	6

Wiederholung

Vollständige Enumeration

Branching – Verzweigen in Teilprobleme

Bounding – Verwerfen von Teilproblemen

Ablauf Branch-and-Bound Verfahren

29 B&B für ein Zuordnungsproblem

B&B mit LP-Relaxierung für ein Rucksackproblem

Zusammenfassung & Ausblick

Wiederholung

Vollständige Enumeration

Branching – Verzweigen in Teilprobleme

Bounding – Verwerfen von Teilproblemen

Lower Bound

Upper Bound und Relaxierung

Ablauf Branch-and-Bound Verfahren

B&B für ein Zuordnungsproblem

B&B mit LP-Relaxierung für ein Rucksackproblem

Wiederholung

Vollständige
Enumeration

Branching –
Verzweigen in
Teilprobleme

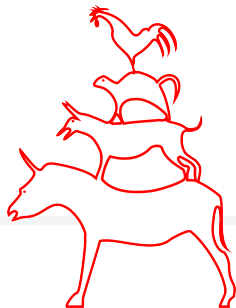
Bounding – Verwerfen
von Teilproblemen

Ablauf
Branch-and-Bound
Verfahren

B&B für ein
Zuordnungsproblem

29 B&B mit
LP-Relaxierung für ein
Rucksackproblem

Zusammenfassung &
Ausblick



In welche Projekte soll investiert werden, um bei einem Investitionsbudget von 25 Mio. € den erwarteten Gewinn zu maximieren?

	Projekt j				
	1	2	3	4	5
erw. Gewinn (Mio. €) u_j	10	9	12	5	9
Investition (Mio. €) w_j	5	6	12	10	12

Formulierung als Rucksackproblem

$$\max \quad 10x_1 + 9x_2 + 12x_3 + 5x_4 + 9x_5$$

$$\text{s.t.} \quad 5x_1 + 6x_2 + 12x_3 + 10x_4 + 12x_5 \leq 25$$

$$x_i \in \{0, 1\} \quad i = 1, \dots, 5$$

Wiederholung

Vollständige
EnumerationBranching –
Verzweigen in
TeilproblemeBounding – Verwerfen
von TeilproblemenAblauf
Branch-and-Bound
VerfahrenB&B für ein
Zuordnungsproblem299 B&B mit
LP-Relaxierung für ein
RucksackproblemZusammenfassung &
Ausblick

▶ **Relaxierung:** LP-Relaxierung

D.h. die Restriktion $x_j \in \{0, 1\}$ wird zu $0 \leq x_j \leq 1$ relaxiert.

▶ **Auswahl** des zu verzweigenden Problems: LIFO-Regel

▶ **Verzweigungs-Variable:** beliebige Variable x_j mit nicht-ganzzahligem Wert. Fixiere einmal $x_j = 0$ und einmal $x_j = 1$.

Üblicherweise: Löse LP-relaxiertes Problem mit Simplex.

Ausnahme: ist es hier einfacher, da **nur eine** Restriktion:

- ▶ Berechne für jedes j den Nutzen pro Gewichtseinheit ($\frac{u_j}{w_j}$)
- ▶ Weise in monoton absteigender Reihenfolge von $\frac{u_j}{w_j}$ den zugehörigen Variablen x_j den Wert 1 zu, solange das Budget nicht erschöpft ist. Plane das erste nicht mehr voll aufnehmbare Projekt anteilig ein.

	Projekt j				
	1	2	3	4	5
u_j	10	9	12	5	9
w_j	5	6	12	10	12
$\frac{u_j}{w_j}$	2	1.5	1	0.5	0.75

Werte

Reihenfolge	1.	2.	3.	4.	5.
Wert Variablen	$x_1 = 1$	$x_2 = 1$	$x_3 = 1$	$x_5 = \frac{2}{12}$	$x_4 = 0$
Ausnutzung Budget	5	11	23	25	25

Liste aktiver Probleme: $L = (P_0)$

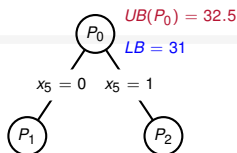
Bearbeitetes Problem: P_0 , wegen LIFO-Regel

- ▶ Löse Relaxierung P_0^R mit $0 \leq x_i \leq 1, \forall i$:

$$x^*(P_0^R) = (1, 1, 1, 0, 1/6) \text{ mit } z^*(P_0^R) = 32.5, \text{ d.h. } UB(P_0) = 32.5$$

- ▶ (Optional:) Lower Bound? Wert von x_5 kann auf 0 abgerundet werden,
 $LB = f(1, 1, 1, 0, 0) = 31$

- ▶ Prüfe Auslotung von P_0 : $UB(P_0) > LB$ (daher nicht **Fall a**) und Lsg $(1, 1, 1, 0, 1/6)$ ist zulässig für P_0^R (daher nicht **Fall c**) aber nicht für P_0 (daher nicht **Fall b**). P_0 ist zu verzweigen und zwar anhand von x_5 .



Liste aktiver Probleme $L = (P_2, P_1)$ (Reihenfolge bei Handrechnung nicht eindeutig!)

	1	2	3	4	5
u_j	10	9	12	5	9
w_j	5	6	12	10	12
u_j/w_j	2	1.5	1	0.5	0.75

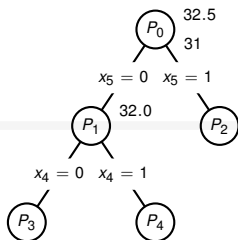
Liste aktiver Probleme: $L = (P_2, P_1)$

Bearbeitetes Problem: P_1 , wegen LIFO

- ▶ Löse Relaxierung P_1^R (P_0 mit $x_5 = 0$):

$x^*(P_1^R) = (1, 1, 1, 1/5, 0)$ mit $z^*(P_1^R) = 32.0$, d.h. $UB(P_1) = 32.0$

- ▶ Prüfe Auslotung: P_1 ist **nicht auslotbar**, weil $UB(P_1) > LB$ und $x^*(P_1^R)$ unzulässig für P_0 . Verzweige P_1 anhand von x_4 .



Liste aktiver Probleme $L = (P_2, P_4, P_3)$

	1	2	3	4	5
u_j	10	9	12	5	9
w_j	5	6	12	10	12
u_j/w_j	2	1.5	1	0.5	0.75

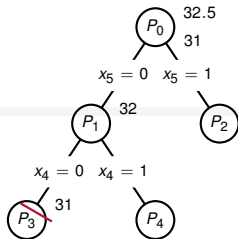
Liste aktiver Probleme: $L = (P_2, P_4, P_3)$

Bearbeitetes Problem: P_3 (P_0 mit $x_5 = 0, x_4 = 0$), wegen LIFO

- ▶ Löse Relaxierung P_3^R :

$$x^*(P_3^R) = (1, 1, 1, 0, 0) \text{ mit } z^*(P_3^R) = 31.0, \text{ d.h. } UB(P_3) = 31$$

- ▶ Prüfe Auslotung: $UB(P_3) \geq LB$ und Lsg $(1, 1, 1, 0, 0)$ ist ganzzahlig (d.h. zulässig für P_0). **Fall b** trifft zu, das Teilproblem P_3 wurde optimal gelöst und ist *auslotbar*. (LB aktualisieren)



	1	2	3	4	5
u_j	10	9	12	5	9
w_j	5	6	12	10	12
u_j/w_j	2	1.5	1	0.5	0.75

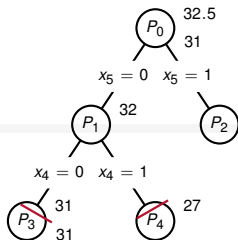
Liste aktiver Probleme: $L = (P_2, P_4)$

Bearbeitetes Problem: P_4 (P_0 mit $x_5 = 0, x_4 = 0$), wegen LIFO-Regel

- ▶ Löse Relaxierung P_4^R :

$$x^*(P_4^R) = (1, 1, 1/3, 1, 0) \text{ mit } z^*(P_4^R) = 27, \text{ d.h. } UB(P_4) = 27.0$$

- ▶ Prüfe Auslotung: $UB(P_4) \leq LB$ gilt, d.h. P_4 ist auszuloten, da es keine bessere Lösung als die bislang beste Lösung enthalten kann (Fall a).



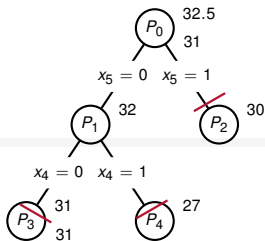
Liste aktiver Probleme: $L = (P_2)$

Bearbeitetes Problem: P_2 (P_0 mit $x_5 = 1$), wegen LIFO-Regel

- ▶ Löse Relaxierung P_2^R :

$$x^*(P_2^R) = (1, 1, 1/6, 0, 1) \text{ mit } z^*(P_2^R) = 30.0, \text{ d.h. } UB(P_2) = 30.0$$

- ▶ Prüfe Auslotung: P_2 ist auslotbar, weil $UB(P_2) < LB$ (Fall a).



Es gibt nun keine aktiven Probleme mehr. Die bislang beste Lösung $x^*(P_3) = (1, 1, 1, 0, 0)$ muss optimal sein!

Fragen zur Wiederholung:

- ▶ Wie wird ein IP/MIP mittels Branch-and-Bound Verfahren gelöst?
- ▶ Was funktioniert die Zerlegung in Teilprobleme (*Branching*) und die Bestimmung einer Untersuchungsreihenfolge der Teilprobleme (*Bounding*), insb. Auslotung?

Klausurrelevante Literatur

Branch-and-Bound-Verfahren Lösung von IP/MIP mit **Branch & Bound**, siehe Domschke/Drexler Kapitel 6.4, 6.5

Ausblick

- ▶ Tutorium: Modellierung **ganzzahliger** Optimierungsprobleme
- ▶ Vorlesung: Heuristische Verfahren zur Lösung von kombinatorischen Optimierungsproblemen, siehe Domschke/Drexler Kapitel 6.2, 6.6.1 oder Nickel/Stein/Waldman Kapitel 6 oder Suhl/Mellouli Kapitel 8.1, 8.2, 8.4

Teil X

Ganzzahlige und kombinatorische Optimierung III – Heuristische Verfahren

Wiederholung

Das kapazitierte Tourenplanungsproblem (CVRP)

Heuristische Lösungsverfahren

Konstruktionsheuristiken

- Nächster-Nachbar-Heuristik für das TSP
- Sukzessives-Einfügen für das TSP
- Savings-Verfahren für das CVRP
- Sweep-Verfahren für das CVRP

Nachbarschaftssuchverfahren



308 Wiederholung

Das kapazitierte
Tourenplanungsproblem (CVRP)

Heuristische
Lösungsverfahren

Konstruktionsheuristiken

Nachbarschaftssuchverfahren

Zusammenfassung &
Ausblick

Branch & Bound Verfahren

Input : Ganzzahliges Problem P_0

- 1 Bestimme LB anhand initialer Lösung (optional)
- 2 Löse die Relaxierung P_0^R von P_0
- 3 **if** P_0 ist ausgelotet **then**
 - 4 | $L := \emptyset$ // L ist eine Liste mit aktiven
| Teilproblemen
- 5 **else**
 - 6 | $L := \{P_0\}$
- 7 **while** $L \neq \emptyset$ **do**
 - 8 | Wähle ein aktives Problem P_k aus L , setze $L := L \setminus \{P_k\}$.
 - 9 | Löse die Relaxierung P_k^R von P_k
 - 10 | **if** P_k ist ausgelotet **then**
 - 11 | | $L := L \setminus \{P_k\}$
 - 12 | **else**
 - 13 | | Verzweige P_k und füge die entstehenden Teilprobleme zu L
| | hinzu.

Output: Optimale Lösung x^* und optimaler Wert LB von P_0

309 Wiederholung

Das kapazitierte
Tourenplanungsproblem (CVRP)Heuristische
Lösungsverfahren

Konstruktionsheuristiken

Nachbarschaftssuchverfahren

Zusammenfassung &
Ausblick

Problem P_i heißt **ausgelotet** und muss nicht verzweigt werden, falls:

- a. $UB(P_i) \leq LB(P_0)$ – Ausloten wg. **Bound**

Der optimale Wert von P_i ist schlechter als der beste bislang gefundene Wert

- b. $UB(P_i) > LB(P_0)$ und $x^*(P_i^R) \in \mathbb{M}(P_0)$ – Ausloten wg. **Optimalität**

Das Teilproblem P_i wurde optimal gelöst, zugleich ist die gefundene Lösung besser als die beste bislang bekannte Lösung.

- c. $\mathbb{M}(P_i^R) = \emptyset$ – Ausloten wg. **Unzulässigkeit**

Für das relaxierte Problem P_i^R gibt es keine zulässige Lösung, daher gibt es auch für P_i keine.

Welches Teilproblem soll als nächstes verzweigt werden?

- ▶ Last-in, first-out (**LIFO-Regel**): das zuletzt der Liste L hinzugefügte Problem wird verzweigt.
- ▶ Maximum-upper-bound (**MUB-Regel**): das aktive Problem P_i mit dem größten $UB(P_i)$ wird verzweigt.

Wie soll ein Problem verzweigt werden? Bei LP-Relaxierung:

- ▶ Verzweige beliebige nicht-ganzzahlige Variable x_i^R , mit $x_i \leq \lfloor x_i^R \rfloor$ bzw. $x_i \geq \lceil x_i^R \rceil$
- ▶ Verzweige Variable mit kleinsten nicht-ganzzahligen Anteil

Wiederholung

Das kapazitierte Tourenplanungsproblem (CVRP)

Heuristische Lösungsverfahren

Konstruktionsheuristiken

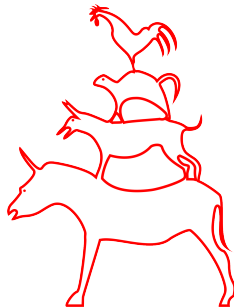
Nächster-Nachbar-Heuristik für das TSP

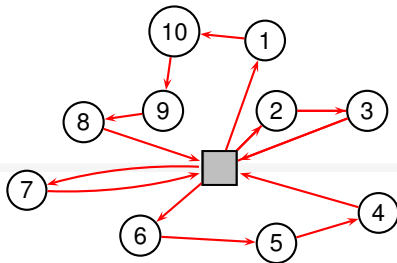
Sukzessives-Einfügen für das TSP

Savings-Verfahren für das CVRP

Sweep-Verfahren für das CVRP

Nachbarschaftssuchverfahren





Depot: Knoten, an dem Fahrzeuge stationiert sind. Im Depot beginnt und endet eine Fahrt.

Tour: Menge aller Kunden, die in derselben Fahrt anzufahren sind.

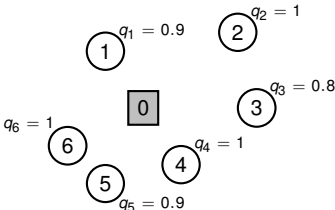
Route: Reihenfolge, in der die Kunden einer Tour zu bedienen sind.

Tourenplan: Menge aller Touren und zugehöriger Routen, die alle Restriktionen erfüllen = zulässige Lösung eines Tourenplanungsproblems.

Das kapazitierte Tourenplanungsproblem

Capacitated Vehicle Routing Problem (CVRP)

- ▶ Knoten $1, \dots, n$ sind **Kunden**(-standorte) i mit einer Nachfrage $q_i > 0$.
- ▶ Im **Depot** (Knoten $0, q_0 = 0$) stehen ausreichend viele **identische** Fahrzeuge mit Fahrzeugkapazität Q bereit.
- ▶ Die Nachfrage eines Kunden ist durch **genau ein** Fahrzeug zu bedienen.
- ▶ Eine **Fahrzeugtour** startet und endet im Depot; die in einer Tour bediente Kundennachfrage darf Q nicht übersteigen.
- ▶ Bediene alle Kunden und **minimiere Gesamtwegstrecke** aller Touren.



Distanzmatrix (c_{ij}):

	0	1	2	3	4	5	6
0	–	20	30	30	20	50	35
1		–	30	45	35	65	45
2			–	30	45	75	55
3				–	35	70	60
4					–	35	25
5						–	25
6							–

Auslieferung von Gütern zu Kunden

- ▶ Büroartikel
- ▶ Auslieferung neuer PKW zum Autohändler
- ▶ Weihnachtsgeschenke durch Weihnachtsmann (zur Not auch DHL, UPS, ...)
- ▶ Essen-auf-Rädern
- ▶ Telekom-Techniker zur Freischalten des DSL-Anschlusses

Einsammeln von Gütern

- ▶ Einsammeln von Milch
- ▶ Entleerung Dixi-Toiletten

Müllabfuhr? Briefträger? Straßenreinigung? → kantenorientierte Modelle

Operations Research

J. Pannek

Wiederholung

314 Das kapazitierte
Tourenplanungsproblem (CVRP)

Heuristische
Lösungsverfahren

Konstruktionsheuristiken

Nachbarschaftssuchverfahren

Zusammenfassung &
Ausblick

- ▶ Heterogener Fuhrpark
- ▶ Zeitfenster, in denen Belieferung erfolgen muss
- ▶ Einsammeln und Ausliefern gleichzeitig (Pickup-and-Delivery)
- ▶ Rechtsabbiegen bevorzugen
- ▶ Minimierung Treibstoffverbrauch
- ▶ Arbeitszeit Fahrer, Lenk- und Ruhezeiten

Operations Research

J. Pannek

Wiederholung

315 Das kapazitierte
Tourenplanungsproblem (CVRP)

Heuristische
Lösungsverfahren

Konstruktionsheuristiken

Nachbarschaftssuchverfahren

Zusammenfassung &
Ausblick

Wiederholung

Das kapazitierte Tourenplanungsproblem (CVRP)

Heuristische Lösungsverfahren

Konstruktionsheuristiken

Nächster-Nachbar-Heuristik für das TSP

Sukzessives-Einfügen für das TSP

Savings-Verfahren für das CVRP

Sweep-Verfahren für das CVRP

Nachbarschaftssuchverfahren



Heuristiken berechnen für ein Optimierungsproblem i.d.R. **gute** aber nicht notwendigerweise optimale Lösungen.

- ▶ Im Gegensatz zu einem Branch-and-Bound-Verfahren
 - ▶ garantiert eine Heuristik weder das Finden einer optimalen Lösung,
 - ▶ noch erfolgt ein Optimalitätsbeweis.
 - ▶ Zudem sind Heuristiken problemspezifisch.
- ▶ Bei Heuristiken handelt es sich lediglich um plausible und erfolgsversprechende Vorgehensregeln.
- ▶ Heuristiken erfreuen sich bei Unternehmen großer Beliebtheit.

Heuristiken berechnen in der Regel
KEINE optimale Lösung!

Fragestellung

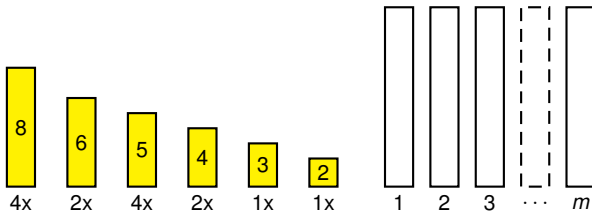
Wie viele Fahrzeuge benötigt man im CVRP mindestens, um alle Kunden zu bedienen?

Gegeben: Eine Menge $J = \{1, \dots, n\}$ von n unteilbaren Objekten mit einer individuellen Größe von w_i , ($i = 1, \dots, n$) sowie m Behältern (bins) mit identischer Kapazität $c \geq w_i$, ($i = 1, \dots, n$).

Gesucht: Die minimale Anzahl von Bins, so dass jedes Objekt aus J in einen Behälter gepackt ist, wobei die Gesamtlänge aller Objekte in einem Bin die Bin-Kapazität nicht übersteigen darf.

Beispiel-Probleminstanz

$c = 12$, $n = 14$, $w = (8, 8, 5, 3, 5, 5, 5, 6, 6, 8, 8, 4, 4, 2)$



Eine Heuristik für das Bin-Packing-Problem

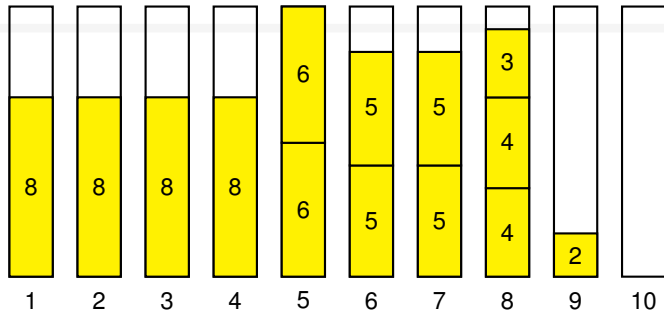
Next Fit Decreasing

Gegeben:

Bins der Länge 12, vierzehn Objekte der Längen 8, 8, 5, 3, 5, 5, 5, 6, 6, 8, 8, 4, 4, 2.

Sortiere Objekte in nicht-aufsteigender Reihenfolge ihrer Länge:

8, 8, 8, 8, 6, 6, 5, 5, 5, 5, 4, 4, 3, 2



Heuristiken garantieren keine optimale Lösung, warum werden sie dennoch eingesetzt?

1. Berechnung von Bounds in Branch-and-Bound-Verfahren
2. Komplizierte Realität wird durch vereinfachtes Modell abgebildet. Ist die optimale Lösung eines **vereinfachten Modells** eine besser Lösung der realen Problemstellung als eine Näherungslösung?
3. Die im Modell verwendeten **Parameterwerte** sind **häufig ungenau** oder unbekannt.
4. Die in der Realität verfügbare **Zeit** zur Berechnung einer optimalen Lösung ist zu kurz. Näherungslösung besser als gar keine Lösung.

Eröffnungsverfahren / Konstruktionsheuristiken

- ▶ Greedy-Heuristiken
- ▶ Vorausschauende Heuristiken

Verbesserungsverfahren / Nachbarschaftssuchverfahren

- ▶ r-opt (2-opt, 3-opt, ...)
- ▶ r-exchange
- ▶ ...

Metaheuristiken

- ▶ Genetische Algorithmen
- ▶ Variable Nachbarschaftssuche
- ▶ Matheuristiken
- ▶ ...

Operations Research

J. Pannek

Wiederholung

Das kapazitierte
Tourenplanungsproblem (CVRP)

320 Heuristische
Lösungsverfahren

Konstruktionsheuristiken

Nachbarschaftssuchverfahren

Zusammenfassung &
Ausblick

Wiederholung

Wiederholung

Das kapazitierte Tourenplanungsproblem (CVRP)

Das kapazitierte
Tourenplanungsproblem (CVRP)

Heuristische Lösungsverfahren

Heuristische
Lösungsverfahren

Konstruktionsheuristiken

Nächster-Nachbar-Heuristik für das TSP

Sukzessives-Einfügen für das TSP

Savings-Verfahren für das CVRP

Sweep-Verfahren für das CVRP

320 Konstruktionsheuristiken

Nächster-Nachbar-
Heuristik für das TSP

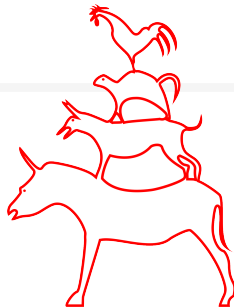
Sukzessives-Einfügen für
das TSP

Savings-Verfahren für das
CVRP

Sweep-Verfahren für das
CVRP

Nachbarschaftssuchverfahren

Zusammenfassung &
Ausblick



Nachbarschaftssuchverfahren

engl. **greedy** = gierig, hier auch kurzfristig

Charakteristika:

Den Entscheidungsvariablen werden **sukzessive** Werte zugewiesen.

Bei jeder Wertzuweisung wird jedesmal die **größtmögliche Verbesserung** (Maximierung) bzw. kleinstmögliche Verschlechterung (Minimierung) der Zielfunktion angestrebt (daher **gierig**).

Einmal erfolgte Wertzuweisung an Variablen werden nicht revidiert.

Vorteil: schnelles und meist einfach nachvollziehbares Verfahren

Nachteil: meist nur mäßige Lösungsgüte

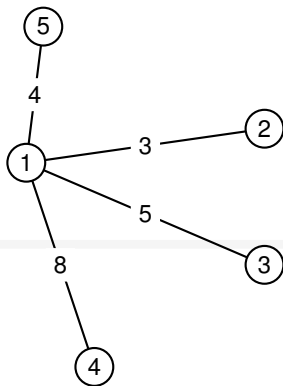
Üblicher Ablauf einer Greedy-Konstruktionsheuristik:

1. Beginne mit einer unzulässigen Lösung x , i.d.R. $x = \emptyset$.
2. Wiederhole 2.1 und 2.2 solange, bis x **zulässig** ist:
 - 2.1 Bewerte jede **Komponente** mit einem Greedy-Kriterium
 - 2.2 Wähle die **gierigste** Komponente (größtmöglichen Lösungsverbesserung) und füge diese x hinzu

Problemspezifische Entwurfsentscheidungen:

Was ist eine Komponente?

Wie misst man die Gier einer Komponente?



Distanzmatrix D für $n = 5$ Städte.

$$D = \begin{pmatrix} 0 & 3 & 5 & 8 & 4 \\ 3 & 0 & 7 & 5 & 2 \\ 5 & 7 & 0 & 10 & 8 \\ 8 & 5 & 10 & 0 & 7 \\ 4 & 2 & 8 & 7 & 0 \end{pmatrix}$$

Ablauf Nächster-Nachbar-Heuristik:

Iteration	Vorläufige Rundreise
1	(1, 2, 1)
2	(1, 2, 5, 1)
3	(1, 2, 5, 4, 1)
4	(1, 2, 5, 4, 3, 1)

Länge gefundene Tour: 27
minimale Tourlänge: 26

Verfahren des nächsten Nachbarn für das TSP

Input : Symmetrisches Traveling Salesman Problem mit n Städten und Distanzmatrix D

```
1 begin
2   Wähle  $s_0 := s_n \in \{1, \dots, n\}$  beliebig als Start und Ziel der
   Rundreise
3   for  $i := 1$  to  $n - 1$  do
4     Bestimme  $s_i$  als diejenige Stadt, die am nächsten an
        $s_{i-1}$  liegt und noch nicht in der Rundreise enthalten
       ist.
```

Output: Rundreise (s_0, s_1, \dots, s_n) mit $s_0 = s_n$

Analogie zu Greedy-Heuristiken:

- ▶ Sukzessive Zuweisung von Werten zu Entscheidungsvariablen.

Abweichung von Greedy-Heuristiken:

- ▶ Die **Konsequenzen** einer Entscheidung, d.h. setzen eines Wertes für eine Variable, auf **zukünftige Entscheidungen** werden abgeschätzt.
- ▶ Entscheidungen können **revidiert** werden, d.h. ein einmal zugewiesener Wert zu einer Entscheidungsvariable kann sich im Verfahrensverlauf wieder ändern.

Vorteil: bessere Lösungen als reines Greedy-Vorgehen

Nachteil: höherer Rechenaufwand als Greedy-Verfahren

Verfahren des sukzessiven Einfügens für das TSP

Input : Symmetrisches Traveling Salesman Problem mit n Städten und Distanzmatrix D

1 **begin**

2 Wähle zwei Städte s_0 und s_1 , die am weitesten voneinander entfernt sind und setze $S := (s_0, s_1, s_0)$

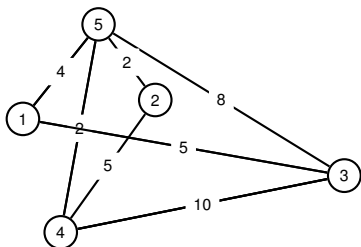
3 **for** $i := 2$ **to** $n - 1$ **do**

4 Bestimme s_i als diejenige Stadt, deren **kleinste Entfernung zu einer Stadt in S** am größten ist.

5 Füge s_i so in S ein, dass die Länge von S am wenigsten zunimmt.

Output: Rundreise S

Sukzessives-Einfügen für das TSP 2/2 – Beispiel



Siehe animierte Folie!
Distanzmatrix wie bei Nächster-Nachbar-Bsp.:

$$D = \begin{pmatrix} 0 & 3 & 5 & 8 & 4 \\ 3 & 0 & 7 & 5 & 2 \\ 5 & 7 & 0 & 10 & 8 \\ 8 & 5 & 10 & 0 & 7 \\ 4 & 2 & 8 & 7 & 0 \end{pmatrix}$$

Iteration	Gewählter Knoten s_i	Einfügetest	Länge
1	$s_2 = 5$	(3,5,4,3) (3,4,5,3)	25 25
2	$s_3 = 1$	(3,1,5,4,3) (3,5,1,4,3) (3,5,4,1,3)	26 29 28
3	$s_4 = 2$	(3,2,1,5,4,3) (3,1,2,5,4,3) (3,1,5,2,4,3) (3,1,5,4,2,3)	31 27 26 28

Operations Research

J. Pannek

Wiederholung

Das kapazitierte
Tourenplanungsproblem
(CVRP)

Heuristische
Lösungsverfahren

Konstruktionsheuristiken

Nächster-Nachbar-
Heuristik für das TSP

32 Sukzessives-Einfügen für
das TSP

Savings-Verfahren für das
CVRP

Sweep-Verfahren für das
CVRP

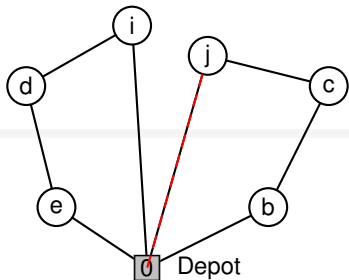
Nachbarschaftssuchverfahren

Zusammenfassung &
Ausblick

Savings-Verfahren — Idee

Berechne die durch Vereinigung von zwei Touren eingesparte Wegstrecke (**saving**)

Einsparung durch Vereinigung der beiden Touren: $s_{ij} = c_{i0} + c_{0j} - c_{ij}$



- ▶ Zwei Touren $(0, e, d, i, 0)$ und $(0, j, c, b, 0)$ mit Kanten $\{0, i\}$ und $\{0, j\}$
- ▶ Entferne Kante $\{0, i\}$ und $\{0, j\}$ und füge Kante $\{i, j\}$ hinzu
- ▶ Kombination der Touren nur möglich, falls Kundenbedarf der Tour \leq Fahrzeugkapazität

1. Berechne Savings-Werte s_{ij} für alle Kundenpaare i, j :

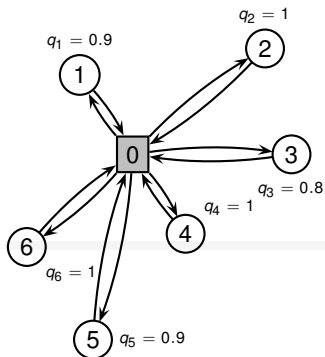
$$s_{ij} = c_{i0} + c_{0j} - c_{ij}$$

Konstruiere Pendeltouren $(0, i, 0)$, $i = 1, \dots, n$

2. Betrachte (und lösche) Kundenpaar i, j mit größtem Saving s_{ij} .
3. Bediene Kunde i und Kunde j in **einer neuen, einzigen** Tour, falls:
 - ▶ Beide Kunden i und j sind **Endkunden verschiedener Touren**. Die eine Tour enthält also die Kante $(0, i)$ und die andere Tour enthält die Kante $(0, j)$.
 - ▶ Gesamtbedarf der neuen Tour $\leq Q$.

Weiter mit Schritt 2, bis alle $s_{ij} > 0$ betrachtet wurden.

Savings-Verfahren für das CVRP 3/3 – Beispiel



Distanzmatrix (c_{ij})

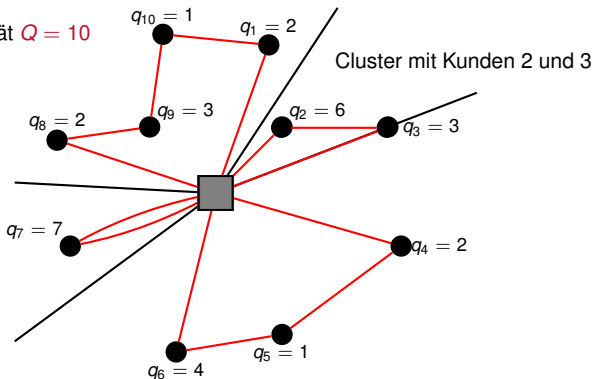
	0	1	2	3	4	5	6
0	—	20	30	30	20	50	35
1		—	30	45	35	65	45
2			—	30	45	75	55
3				—	35	70	60
4					—	35	25
5						—	25
6							—

Savingsmatrix (s_{ij})

Fahrzeugkapazität $Q = 3$

Gesamtlänge der Touren: 370, Anzahl der Touren: 6

Fahrzeugkapazität $Q = 10$



1. Wähle beliebigen Startknoten.
2. Füge nächsten Kunden (entgegen Uhrzeigersinn) zu Cluster hinzu, falls Kapazität $\leq Q$. Beginne andernfalls neuen Cluster.
3. Löse für Knoten jedes Clusters ein TSP, z.B. mit Nächster-Nachbar Heuristik oder Branch-and-Bound.

Wiederholung

Das kapazitierte Tourenplanungsproblem (CVRP)

Heuristische Lösungsverfahren

Konstruktionsheuristiken

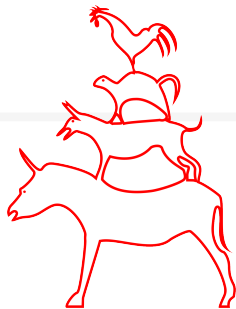
Nächster-Nachbar-Heuristik für das TSP

Sukzessives-Einfügen für das TSP

Savings-Verfahren für das CVRP

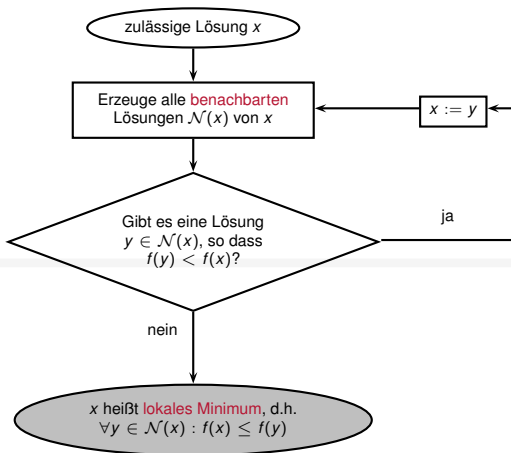
Sweep-Verfahren für das CVRP

Nachbarschaftssuchverfahren



Nachbarschaftssuche für ein Minimierungsproblem 1/2

Begriff des lokalen Minimums



- ▶ **Synonyme:** Lokale Suche oder Verbesserungsverfahren
- ▶ **Voraussetzung:** eine bekannte **zulässige** Lösung $x = (x_1, \dots, x_n)$
- ▶ Alle Lösungen, die durch Anwendung **eines Zugs** auf x entstehen, heißen **Nachbarschaft** $\mathcal{N}(x)$ von x .
- ▶ Ein **Zug (engl. move)** ist eine Vorschrift, wie die Werte von (x_1, \dots, x_n) zu ändern sind.
- ▶ Ein Zug ändert nur wenige Werte von (x_1, \dots, x_n) , daher nennt man die entstehenden Lösungen Nachbarn von x .
- ▶ Gibt es in der Nachbarschaft $\mathcal{N}(x)$ von x eine bessere Lösung, wird diese als neue Ausgangslösung gewählt.
- ▶ Gibt es keine bessere Lösung, dann heißt x **lokales Minimum** und die Suche stoppt.

$$\max f(x) = 3x_1 + 6x_2 + 3x_3 + 4x_4 + 2x_5$$

$$\text{s.t. } x_1 + 4x_2 + 2x_3 + 3x_4 + 2x_5 \leq 8 \quad x_i \in \{0, 1\} \quad i = 1, \dots, 5$$

Die zulässigen Startlösung sei $x = (1, 0, 1, 1, 0)$.

Zug A: Ändere den Wert genau einer binären Variable

	x_1	x_2	x_3	x_4	x_5	$f(x)$	
Ausgangslösung x	1	0	1	1	0	10	
x^1	0	0	1	1	0	$10 - 3 = 7$	
x^2	1	1	1	1	0	$10 + 6 = 16$	unzulässige Lsg!
x^3	1	0	0	1	0	$10 - 3 = 7$	
x^4	1	0	1	0	0	$10 - 4 = 6$	
x^5	1	0	1	1	1	$10 + 2 = 12$	

Die Lösung x hat folgende **Nachbarschaft**: $\mathcal{N}(1, 0, 1, 1, 0) = \{x^1, x^3, x^4, x^5\}$.

Ein einfacher Zug für das Rucksackproblem 2/2

$$\max f(x) = 3x_1 + 6x_2 + 3x_3 + 4x_4 + 2x_5$$

$$\text{s.t. } x_1 + 4x_2 + 2x_3 + 3x_4 + 2x_5 \leq 8 \quad x_i \in \{0, 1\} \quad i = 1, \dots, 5$$

Die zulässigen Startlösung sei $x = (1, 0, 1, 1, 0)$.

Zug B: Tausche einen Gegenstand mit $x_i = 1$ gegen einen mit $x_j = 0$

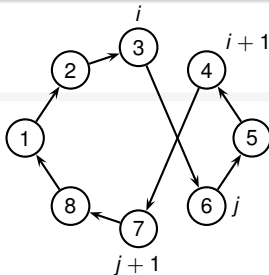
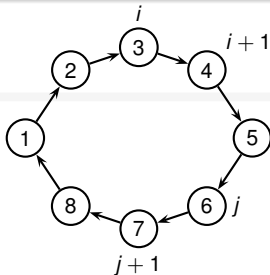
	x_1	x_2	x_3	x_4	x_5	$f(x)$	
Ausgangslösung x	1	0	1	1	0	10	
x^1	0	1	1	1	0	10-3+6=13	unzulässige Lsg!
x^2	0	0	1	1	1	10-3+2=9	
x^3	1	1	0	1	0	10-3+6=13	
x^4	1	0	0	1	1	10-3+2=9	
x^5	1	1	1	0	0	10-4+6=12	
x^6	1	0	1	0	1	10-4+2=8	

Die Lösung x hat folgende **Nachbarschaft**:

$$\mathcal{N}(1, 0, 1, 1, 0) = \{x^2, x^3, x^4, x^5, x^6\}.$$

2-opt Zug

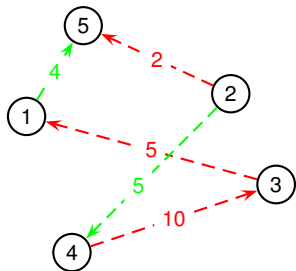
1. Wähle zwei beliebige Pfeile, z.B. $(i, i + 1)$ und $(j, j + 1)$
2. Bilde daraus zwei neue Pfeile, nämlich (i, j) und $(i + 1, j + 1)$ (Endknoten des 1. Pfeils wird zu Startknoten des 2. Pfeils)
3. Kehre die Richtung aller Pfeile zwischen Knoten $i + 1$ und Knoten j um.



Pfeilpaar $(3,4)$ und $(6,7)$ wird zu $(3,6)$ und $(4,7)$

Umkehrung Pfeilrichtungen, d.h. $(4,5) \rightarrow (5,4)$, $(5,6) \rightarrow (6,5)$

Die 2-opt Nachbarschaft für das TSP 2/2



Distanzmatrix D wie
Nächster-Nachbar-Beispiel:

$$D = \begin{pmatrix} 0 & 3 & 5 & 8 & 4 \\ 3 & 0 & 7 & 5 & 2 \\ 5 & 7 & 0 & 10 & 8 \\ 8 & 5 & 10 & 0 & 7 \\ 4 & 2 & 8 & 7 & 0 \end{pmatrix}$$

Nächster-Nachbar-Heuristik ergibt Tour (1, 2, 5, 4, 3, 1) als Startlösung für 2-opt.

Iteration	i	j	Änderung Länge	Rundreise
1	1	3	$4 + 5 - 3 - 7 = -1$	(1, 5, 2, 4, 3, 1)
2	1	3	$3 + 7 - 4 - 5 = 1$	unverändert
	1	4	$8 + 8 - 10 - 4 = 2$	unverändert
	1	5	$5 + 4 - 4 - 5 = 0$	unverändert
	2	4	$7 + 7 - 2 - 10 = 2$	unverändert
	2	5	$8 + 3 - 2 - 5 = 4$	unverändert
	3	5	$7 + 8 - 5 - 5 = 5$	unverändert

Eine Heuristik zur Lösung des CVRP könnte aus folgenden Bausteinen bestehen:

Konstruktion einer ersten zulässigen Lösung:

- ▶ **Savings**-Verfahren
- ▶ Für jede Tour: Versuche die Länge der Routen mit der **Nächster-Nachbar-Heuristik** zu verkürzen.

Nachbarschaftssuche zur Verbesserung der konstruierten Lösung:

- ▶ Für jede Tour: Anwendung von **2-opt** bis lokales Minimum erreicht.

Weitere Verbesserung im Rahmen der Nachbarschaftssuche:
Vertausche Kunden zwischen zwei Touren

Umfangreiche Probleme werden häufig durch Kombination mehrere Basisheuristiken bearbeitet.

Fragen zur Wiederholung:

- ▶ Wie funktionieren Konstruktionsheuristiken und Nachbarschaftssuche?
- ▶ Was bedeuten die Begriffe Lokales Minimum, Nachbarschaft, Nächster-Nachbar, Sukzessives Einfügen (TSP), Savings und Sweep (CVRP)?

Klausurrelevante Literatur

Heuristische Verfahren zur Lösung von kombinatorischen Optimierungsproblemen, siehe Domschke/Drexler Kapitel 6.2, 6.6.1 oder Nickel/Stein/Waldman Kapitel 6 oder Suhl/Mellouli Kapitel 8.1, 8.2, 8.4

Ausblick

- ▶ Tutorium: Heuristische Verfahren
- ▶ Vorlesung: Simulation, siehe Domschke/Drexler Kapitel 10 oder Werners Kapitel 7

Wiederholung

Grundlagen der
Simulation

Deterministische
Simulation zur
Maschinenbelegung

Stochastische
Simulation

Stochastische
Simulation zur
Risikoanalyse von
Investitionsprojekten

Zusammenfassung &
Ausblick

Teil XI

Simulation

Wiederholung

Grundlagen der Simulation

Wesen

Arten der Simulation

Deterministische Simulation zur Maschinenbelegung

Stochastische Simulation

Verteilungsfunktion

Erzeugung von Zufallszahlen

Stochastische Simulation zur Risikoanalyse von Investitionsprojekten



340 Wiederholung

Grundlagen der Simulation

Deterministische Simulation zur Maschinenbelegung

Stochastische Simulation

Stochastische Simulation zur Risikoanalyse von Investitionsprojekten

Zusammenfassung & Ausblick

Grundstrukturen

- ▶ Lineares Problem
- ▶ Ganzzahlige Variablen
- ▶ Logische/Mengenmäßige Verknüpfungen

Standardprobleme

- ▶ Zuordnungsproblem
- ▶ Reihenfolgeproblem
- ▶ Gruppierungsproblem
- ▶ Auswahlproblem

Grundlagen der
Simulation

Deterministische
Simulation zur
Maschinenbelegung

Stochastische
Simulation

Stochastische
Simulation zur
Risikoanalyse von
Investitionsprojekten

Zusammenfassung &
Ausblick

- ▶ Branch-and-Bound Verfahren
- ▶ Nächster-Nachbar-Heuristik
- ▶ Sukzessives-Einfügen
- ▶ Savings-Verfahren
- ▶ Sweep-Verfahren

343 **Wiederholung**

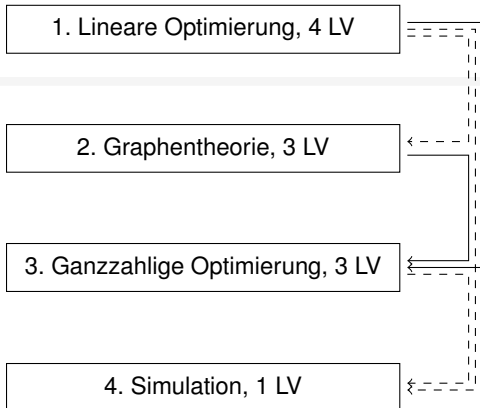
Grundlagen der Simulation

Deterministische Simulation zur Maschinenbelegung

Stochastische Simulation

Stochastische Simulation zur Risikoanalyse von Investitionsprojekten

Zusammenfassung & Ausblick



Wiederholung

Grundlagen der Simulation

Wesen

Arten der Simulation

Deterministische Simulation zur Maschinenbelegung

Stochastische Simulation

Verteilungsfunktion

Erzeugung von Zufallszahlen

Stochastische Simulation zur Risikoanalyse von Investitionsprojekten



Wiederholung

343 Grundlagen der Simulation

Wesen

Arten der Simulation

Deterministische Simulation zur Maschinenbelegung

Stochastische Simulation

Stochastische Simulation zur Risikoanalyse von Investitionsprojekten

Zusammenfassung & Ausblick

Idee:

- ▶ Modellieren von Prozessen realer Systeme
- ▶ Durchführen von (Stichproben-)Experimenten

Zweck:

- ▶ Systemverhalten erklären oder gestalten, insbesondere falls es von umfangreichen stochastischen Einflüssen geprägt ist

Beispiele für den Einsatz von Simulation

Windkanal Automobilbau

Flugsimulator zur Pilotenausbildung

Tierversuche zur Untersuchung der Wirkung von Medikamenten

Mathematisches Modell eines Lagerhauses oder Produktionssystems

- ▶ Ein vollständiges mathematisches Optimierungsmodell ist nicht verfügbar oder die Kosten zur Entwicklung sind zu hoch
- ▶ Analytische Methoden, sofern vorhanden, erfordern zu stark vereinfachende Annahmen, sodass der Kern des Problems verfälscht wird
- ▶ Analytische Methoden zu kompliziert bzw. so aufwändig, dass Einsatz nicht praktikabel erscheint
- ▶ Reale Experimente, z.B. mit Prototypen (Flugzeug, Lagersystem), zu kostspielig
- ▶ Beobachtung reales System zu gefährlich (Reaktorverhalten) oder zu zeitaufwändig (Konjunkturschwankungen)

	statisch	dynamisch	
geschlossen			keine Umweltbeziehung
offen			mit Umweltbeziehung
	unveränderlicher Systemzustand	Systemzustand ändert sich	

Besonders relevant: offene & dynamische Systeme (**Warteschlangen, Produktion, Warenhaus ...**)

Deterministische Simulationsmodelle

Eingangsgrößen und Parameter bestimmen eindeutig Ausgangsgrößen.

Stochastische Simulationsmodelle

Zufallseinflüsse treten im System auf, welche durch Zufallsvariablen mit Verteilungsfunktionen charakterisierbar sind.

Name: Roulette in einem Spielkasino in Monte Carlo

Eigenschaften des Roulette und der Monte-Carlo-Simulation:

- ▶ Wahrscheinlichkeit für eine bestimmte Zahl a priori bekannt und für alle Zahlen gleich groß
- ▶ Wahrscheinlichkeit für eine bestimmte Zahl ist unabhängig von vorherigen Würfeln der Kugel

Anwendung:

- ▶ Lösung von Integralen oder komplizierten Integral- und Differentialgleichungen
- ▶ Künstliche Erzeugung von Stichproben

Mit Monte-Carlo-Simulation lassen sich auch deterministische Probleme lösen!

Eigenschaften:

- ▶ Zustand wird durch zeitabhängige Variablen beschrieben
- ▶ Zustandsvariablen ändern sich bei Eintritt bestimmter Ereignisse

Typen:

- ▶ Periodenorientierte Zeitführung
- ▶ Ereignisorientierte Zeitführung

Eigenschaften:

- ▶ Zustandsvariablen ändern sich kontinuierlich mit der Zeit
- ▶ Modellierung typischerweise mit Differentialgleichungen

Beispiel diskrete vs. kontinuierliche Modelle

- ▶ Geschwindigkeit eines PKW ändert sich kontinuierlich bei Druck auf Gaspedal
- ▶ Kontinuierlicher Verkehrsfluss an einer Ampel aus Vogelperspektive
- ▶ Diskreter Verkehrsfluss an einer Ampel aus Fußgängerperspektive
- ▶ Diskrete Warteschlange bei Essensausgabe in einer Mensa

Wiederholung

Wiederholung

Grundlagen der Simulation

Grundlagen der Simulation

Wesen

349 **Deterministische Simulation zur Maschinenbelegung**

Arten der Simulation

Stochastische Simulation

Deterministische Simulation zur Maschinenbelegung



Stochastische Simulation zur Risikoanalyse von Investitionsprojekten

Stochastische Simulation

Verteilungsfunktion

Erzeugung von Zufallszahlen

Zusammenfassung & Ausblick

Stochastische Simulation zur Risikoanalyse von Investitionsprojekten

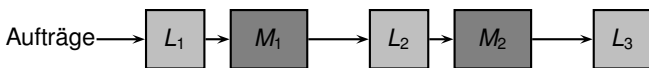
Es gibt zwei Maschinen M_1 , M_2 und vier Aufträge A_1 , A_2 , A_3 , A_4 .

Bearbeitungsdauern von Aufträgen auf Maschinen:

Maschine	Auftrag			
	A_1	A_2	A_3	A_4
M_1	2	1	3	4
M_2	4	2	1	2

Ein Auftrag A_i muss zuerst mit M_1 und danach mit M_2 bearbeitet werden.

Lager L_i vor und nach jeder Maschine

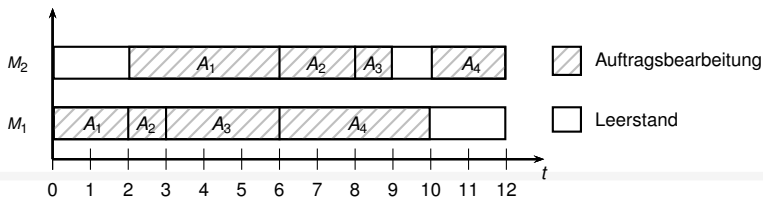


Mittels Simulation soll das dynamische Verhalten des Produktionssystems untersucht werden:

- ▶ **Variable Eingangsgrößen:** Reihenfolge, in der die Aufträge zu bearbeiten sind, z.B. A_1, A_2, A_3, A_4
- ▶ **Parameter:** Bearbeitungsdauer, z.B. Tabelle vorherige Folie
- ▶ **Ausgangsgröße:** Zeitdauer Gesamtdurchlauf, Auslastung Maschinen, durchschnittliche Wartezeit eines Auftrags vor einer Maschine

Periode	L ₁	M ₁	L ₂	M ₂	L ₃
Initialz.	A ₁ , A ₂ , A ₃ , A ₄				
1	A ₂ , A ₃ , A ₄	A ₁			
2	A ₂ , A ₃ , A ₄	A ₁			
3	A ₃ , A ₄	A ₂		A ₁	
4	A ₄	A ₃	A ₂	A ₁	
5	A ₄	A ₃	A ₂	A ₁	
6	A ₄	A ₃	A ₂	A ₁	
7		A ₄	A ₃	A ₂	A ₁
8		A ₄	A ₃	A ₂	A ₁
9		A ₄		A ₃	A ₁ , A ₂
10		A ₄			A ₁ , A ₂ , A ₃
11				A ₄	A ₁ , A ₂ , A ₃
12				A ₄	A ₁ , A ₂ , A ₃
Endez.					A ₁ , A ₂ , A ₃ , A ₄
Dauer	11	10	5	9	13

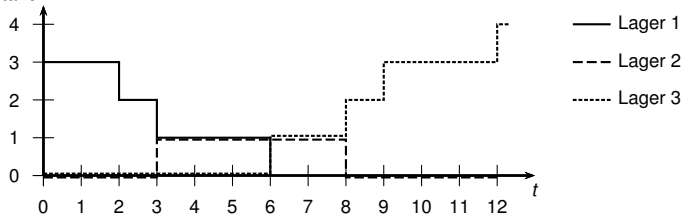
Gesamtdurchlaufzeit: 12 Perioden



Auslastung $M_1: \frac{10}{12} = 83\frac{1}{3}\%$ und $M_2: \frac{9}{12} = 75\%$

Wartezeit: vor $M_1: \frac{11}{4} = 2.75$ Perioden; $M_2: \frac{5}{4} = 1.25$

Bestand



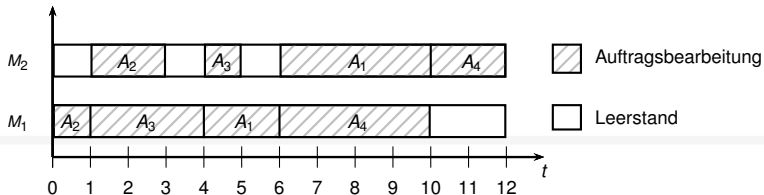
Aufgabe

Maschinen M_1 , M_2 und Aufträge A_1 , A_2 , A_3 , A_4 wie bisher. Die Bearbeitungsreihenfolge der Aufträge erfülle jedoch nun:

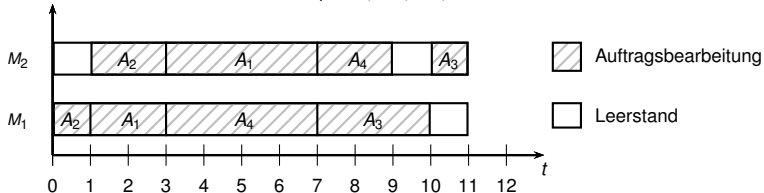
Der Auftrag mit der kürzesten Bearbeitungsdauer (Operationszeit) wird als nächstes bearbeitet (KOZ-Regel).

Skizzieren Sie ein Maschinenbelegungsdiagramm und geben sie die Auslastung von M_1 und M_2 an.

Bearbeitungsreihenfolge: Der Auftrag mit der jeweils kürzesten Gesamtbearbeitungsdauer wird zuerst bearbeitet.



Auftragsreihenfolge: minimale Gesamtdurchlaufzeit (mittels Branch-and-Bound bestimmt): A_2, A_1, A_4, A_3



- ▶ Simulation eignet sich vor allem zur **Erklärung bzw. zur Analyse** von Systemen (Analyse-/Erklärungsmodelle).
- ▶ Interessiert lediglich die Bearbeitungsreihenfolge mit der kürzesten Durchlaufzeit, sollten Verfahren der ganzzahligen Optimierung verwendet werden.
- ▶ Komplexität realitätsnäherer Maschinenbelegungsprobleme deutlich höher als dieses Beispiel
- ▶ In der Praxis meist konkurrierende Zielfunktionen:
 - ▶ min Durchlaufzeit
 - ▶ max Termineinhaltung (min maximale Terminüberschreitung)
 - ▶ max Kapazitätsauslastung
 - ▶ min Zwischenlagerbestand

Wiederholung

Grundlagen der Simulation

Wesen

Arten der Simulation

Deterministische Simulation zur Maschinenbelegung

Stochastische Simulation

Verteilungsfunktion

Erzeugung von Zufallszahlen

Stochastische Simulation zur Risikoanalyse von Investitionsprojekten



Operations Research

J. Pannek

Wiederholung

Grundlagen der
Simulation

Deterministische
Simulation zur
Maschinenbelegung

356 Stochastische
Simulation

Verteilungsfunktion
Erzeugung von
Zufallszahlen

Stochastische
Simulation zur
Risikoanalyse von
Investitionsprojekten

Zusammenfassung &
Ausblick

Zufallsgeschehen des realen Systems wird durch Zufallsvariable erfasst, z.B.

- ▶ Ausfallhäufigkeit von Maschinen
- ▶ Eingang neuer Aufträge
- ▶ Zeitpunkt und Dauer von Anrufen im Call Center

Beschreibung des Verhaltens einer Zufallsvariable X mittels:

$$\text{Dichtefunktion: } f(x) = P(X = x) = \begin{cases} p_i & \text{falls } X = x \\ 0 & \text{sonst} \end{cases}$$

$$\text{Verteilungsfunktion: } F(x) = P(X \leq x) = \sum_{x_i \leq x} f(x_i)$$

1. Wie kann das Verteilungsgesetz ermittelt werden, welches das Verhalten einer Zufallsvariable beschreibt?
2. Wie können entsprechende Zufallswerte erzeugt werden?

Verteilungsfunktion von Zufallsvariablen i.A. unbekannt, daher zwei Vorgehensweisen:

1. Zusatzinformationen rechtfertigen die Annahme einer bestimmten theoretischen Verteilung.

Bsp.: die Lebensdauer von Bauteilen meist **exponentialverteilt**.

Lediglich Verteilungsparameter aus Stichprobe ableiten, z.B. Erwartungswert, Standardabweichung bei Normalverteilung.

2. Ohne Zusatzinformationen Beschränkung auf **empirische Verteilung**.

Statistische **Hypothesentests** zur

- ▶ Überprüfung der getroffenen Annahmen über eine Verteilung sowie
- ▶ Ermittlung von Konfidenzintervallen von Verteilungsparametern.

Beobachtungswerte können näherungsweise mit einer **empirischen Verteilungsfunktion** beschrieben werden.

1. Erfasse Stichprobenwerte in Klassen
2. Ermittle Klassenhäufigkeiten
3. Berechne Verteilungsfunktion

Ausfallverhalten eines Zahnkranzpaars in einer Maschine

Laufzeit: Zeit von Einbau des Zahnkranzpaars bis Ausfall

Klassen: bis 3749h, von 3750h bis 4249h, von 4250h bis 4749h, ...

Laufzeit in 1000h	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8
Ausfälle	200	400	600	800	600	520	480	200	120	80
rel. Häufigkeit	0.05	0.10	0.15	0.20	0.15	0.13	0.12	0.05	0.03	0.02
Verteilungsfkt.	0.05	0.15	0.30	0.50	0.65	0.78	0.90	0.95	0.98	1.00

Ein Zahnkranzpaar hat mit einer Wahrscheinlichkeit von 30 Prozent eine Laufzeit von weniger als 4749 Stunden.

Echte Zufallszahl: Ziehen von Kugeln aus einer Urne, Würfeln

Unechte oder **Pseudo-Zufallszahl**:

- ▶ Ausnutzen von unregelmäßigen Ziffernfolgen von π , $\sqrt{2}$ oder e
- ▶ **Arithmetischer Zufallszahlengenerator**

Linearer Kongruenzgenerator nach Lehmer

$$x_{i+1} = (a \cdot x_i + b) \bmod m$$

Die $(i + 1)$ -te **Zufallszahl** x_{i+1} wird aus i -ten Zufallszahl x_i erzeugt.

Vorgegebene **Parameter**: a, b, m ganzzahlig mit $a, m > 0$ und $b \geq 0$ sowie $a, b < m$; ein Startwert x_0 (**Saatwert**, engl. **seed**)

Der Operator **mod** gibt den Rest bei Ganzzahldivision an, z.B. $10 \bmod 3 = 1$ ($10/3 = 3$ **Rest 1**) oder $22 \bmod 6 = 4$ ($22/6 = 3$ **Rest 4**).

Berechnung einer **Standardzufallszahl** z_i , die im Intervall $(0, 1)$ gleichverteilt ist: $z_i = \frac{x_i}{m-1}$.

Zufallszahlengenerator: $x_{i+1} = (a \cdot x_i + b) \bmod m$

1. Variante: $a = 4, b = 0, m = 5$, Saatwert $x_0 = 4$

$x_0 = 4, x_1 = (4 \cdot 4) \bmod 5 = 1, x_2 = (4 \cdot 1) \bmod 5 = 4, x_3 = (4 \cdot 4) \bmod 5 = 1, \dots$ Periodenlänge ist 2

2. Variante: $a = 3, b = 2, m = 8$, Saatwert $x_0 = 7$

$x_0 = 7, x_1 = (3 \cdot 7 + 2) \bmod 8 = 7, \dots \rightarrow$ Periodenlänge ist 1

Aufgabe

Wie lautet die generierte Zufallszahlensequenz mit Startwert $x_0 = 4$ und Parametern $a = 5, b = 3, m = 8$?

$x_0 = 4, x_1 = (5 \cdot 4 + 3) \bmod 8 = 7, x_2 = (5 \cdot 7) \bmod 8 = 6,$

$x_3 = 1, x_4 = 0, x_5 = 3, x_6 = 2, x_7 = 5, x_8 = 4, \dots$

\rightarrow Periodenlänge ist 8 ($= m$) und damit maximal.

- ▶ Möglichst genaue Annäherung der Pseudo-Zufallszahlen an die gewünschte Verteilungsfunktion
- ▶ Zufallszahlenfolge soll reproduzierbar sein, um (Computer-)Experimente mit Zufallszahlen reproduzieren zu können
- ▶ Effiziente Berechnung
- ▶ Zahlenfolge soll **große Periodenlänge** aufweisen
- ▶ Zufallszahlenfolge soll im Intervall $(0, 1)$ eine hohe Besetzungsdichte haben, d.h. durchschnittliche und maximale Abstände zwischen den Zahlen sollen möglichst gering sein.

Ableitung von Zufallszahlen

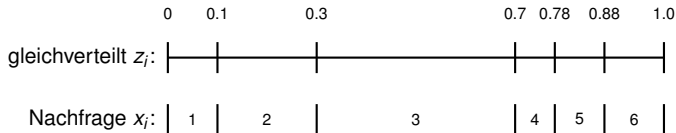
Zufallszahlen, die beliebigen Verteilungen genügen, lassen sich aus (0,1)-Zufallszahlen ableiten (z.B. Werners Kapitel 7.2.2 oder Domschke/Drechsler Kapitel 10.3)

Transformation einer empirischen Verteilung:

Nachfragemenge	1	2	3	4	5	6
Häufigkeit ($\sum = 50$)	5	10	20	4	5	6
relative Häufigkeit	0.1	0.2	0.4	0.08	0.1	0.12

Unterteilung des Intervalls (0, 1) gemäß relativer Häufigkeiten in disjunkte Abschnitte:

$]0, 0.1[$, $]0.1, 0.3[$, $]0.3, 0.7[$, $]0.7, 0.78[$, $]0.78, 0.88[$, $]0.88, 1.0[$



Wiederholung

Grundlagen der Simulation

Deterministische Simulation zur Maschinenbelegung

Stochastische Simulation

Verteilungsfunktion

36 Erzeugung von Zufallszahlen

Stochastische Simulation zur Risikoanalyse von Investitionsprojekten

Zusammenfassung & Ausblick

Wiederholung

Grundlagen der Simulation

Wesen

Arten der Simulation

Deterministische Simulation zur Maschinenbelegung

Stochastische Simulation

Verteilungsfunktion

Erzeugung von Zufallszahlen

Stochastische Simulation zur Risikoanalyse von Investitionsprojekten

Wiederholung

Grundlagen der
Simulation

Deterministische
Simulation zur
Maschinenbelegung

Stochastische
Simulation

364 Stochastische
Simulation zur
Risikoanalyse von
Investitionsprojekten

Zusammenfassung &
Ausblick

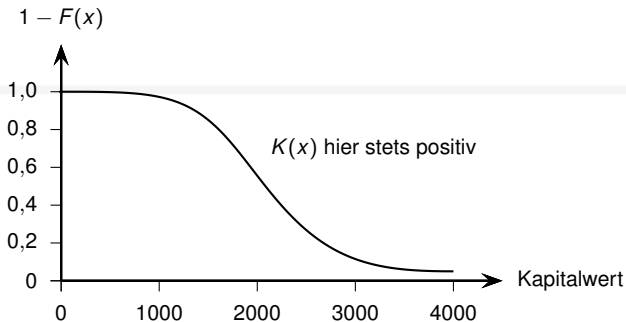


Vorteilhaftigkeit eines Investitionsprojektes mittels Kapitalwert
Kapitalwert: Abzinsung aller Ein- und Auszahlungen auf den
Startzeitpunkt der Investition

$$K(x) = \sum_{t=0}^T x_t \cdot (1 + i)^{-t}$$

- ▶ i Kalkulationszins
- ▶ T Laufzeit (Perioden) des Projekts
- ▶ x_t Einzahlungen minus Auszahlungen in Periode t
- ▶ Interpretation:
 - $K(x) > 0 \rightarrow$ Projekt lohnenswert
 - $K(x) < 0 \rightarrow$ Projekt **nicht** lohnenswert

- ▶ Zahlungen x_t unterliegen zufälligen Einflüssen
- ▶ Verschiedene Zahlungsreihen werden ermittelt, z.B. durch Expertenbefragungen
- ▶ $F(x)$ ist Verteilungsfunktion des unsicheren Kapitalwertes $K(x)$
- ▶ $1 - F(x)$ ist **Risikoprofil**



Aufgabe: Ermittlung des Risikoprofils für Investitionsprojekt:

Zeitpunkt	$t = 0$	$t = 1$	$t = 2$	$t = 3$
Zahlung	-2100	[900;1100]	[850;1150]	[830;1230]

Rückzahlung in $t = 1$ zufällig (gleichverteilt) im Intervall [900; 1100]

Annahmen:

- ▶ Kalkulationszins $i = 10\%$
- ▶ Zufallszahlungen jeder Periode sind unabhängig voneinander
- ▶ Zufallszahlungen sind im angegebenen Intervall gleichverteilt

Frage: Wie kann man Simulation zur Ermittlung eines Risikoprofils einsetzen?

Vorgehensweise:

1. Berechne Zufallszahlen und leite $[0, 1[$ gleichverteilte Zufallszahlen ab
2. Bestimme Transformationsfunktion für gegebene Intervalle
3. Berechne Kapitalwert für jede Zufallszahl

Beispiel

zu 1) Zufallszahlengenerator: $u_k = (5u_{k-1} + 3) \bmod 64$, $u_0 = 4$

Es sollen 10 Realisationen betrachtet werden, dazu benötigt man 30 Zufallszahlen: 4, 23, 54, 17, 24, 59, 42, 21, 44, 31, ...

$[0, 1[$ -Zufallszahlen (gerundet):

0.06; 0.37; 0.86; 0.27; 0.38; 0.94; 0.67; 0.33; 0.70; 0.49; ...

zu 2) Transformationsfunktionen:

$$[900; 1100] \rightarrow z_1(u) = 900 + 200u$$

$$[850; 1150] \rightarrow z_2(u) = 850 + 300u$$

$$[830; 1230] \rightarrow z_3(u) = 830 + 400u$$

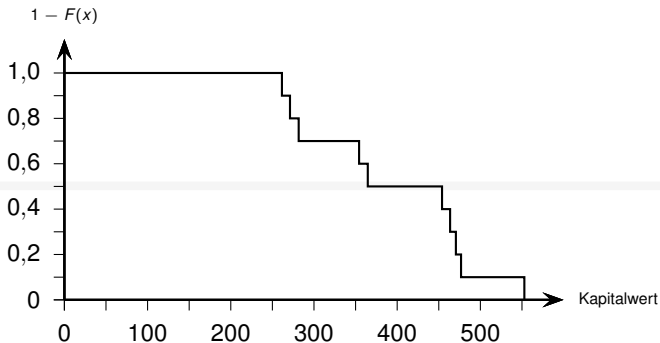
Beispiel (Fortsetzung) – zu 3) Zahlungen und Kapitalwerte

Realisation k	x_1	x_2	x_3	$K^k(x)$
1	912.70	992.86	1083.97	364.67
2	973.02	969.05	899.84	261.49
3	1071.43	850.00	1198.25	476.77
4	593.97	864.29	1064.92	281.62
5	976.19	935.71	1210.95	470.57
6	1087.30	988.10	1128.41	552.86
7	1033.33	945.24	1122.06	463.61
8	966.67	1035.71	1090.32	453.92
9	1039.68	878.57	931.59	271.17
10	998.41	1007.14	950.63	354.22

$$K^1 = -2100 + 912.70 \cdot 1.1^{-1} + 992.86 \cdot 1.1^{-2} + 1083.97 \cdot 1.1^{-3} = 364.67$$

$$K^2 = -2100 + 973.02 \cdot 1.1^{-1} + 969.05 \cdot 1.1^{-2} + 899.84 \cdot 1.1^{-3} = 261.49$$

k	2	9	4	10	1	8	7	5	3	6
K^k	261.49	271.17	281.62	354.22	364.67	453.92	463.61	470.57	476.77	552.86
$F(x)$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$1 - F(x)$	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0



Risikoprofil kann nun mittels statistischem und finanzwirtschaftlichen Instrumentarium eingehender analysiert werden.

Wiederholung

Grundlagen der Simulation

Deterministische Simulation zur Maschinenbelegung

Stochastische Simulation

376 Stochastische Simulation zur Risikoanalyse von Investitionsprojekten

Zusammenfassung & Ausblick

Fragen zur Wiederholung:

- ▶ Wie funktioniert Maschinenbelegung mittels **deterministischer Simulation**?
- ▶ Was ist eine **stochastische Simulation** und wie wird sie zur Risikioanalyse eingesetzt?

Klausurrelevante Literatur

Domschke/Drexler Kapitel 10 oder Werners Kapitel 7

Ausblick

- ▶ Tutorium: Simulation

Teil XII

Kooperative Spieltheorie

Motivation und Problemstellung

Kooperative Spiele

Grundbegriffe

Gewichtete Abstimmungsspiele

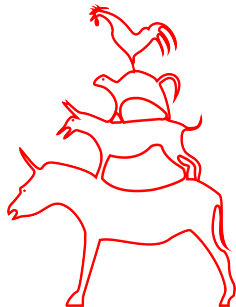
Der Shapley-Wert als ein Lösungskonzept

Punktwertige Lösungskonzepte

Shapley Axiome

Die Shapley-Formel

Anwendung der Shapley-Formel





Drei Personen mit verschiedenen Fahrtzielen teilen sich ein Taxi. Wie sollen die anfallenden Kosten auf die drei Personen verteilt werden?



Kunden der kooperierenden Banken können an Geldautomaten kostenlos Bargeld abheben.

Geldautomatennetze der Mitglieder stark heterogen (regionale Verteilung, Anzahl)

Wie soll der Kooperations*nutzen* auf die Mitglieder einer Kooperation verteilt werden?

Operations Research

J. Pannek

37 Motivation und Problemstellung

Kooperative Spiele

Der Shapley-Wert als ein Lösungskonzept

Zusammenfassung & Ausblick

Beispiel: Wasserwerk für zwei Städte

Ein Kostenaufteilungsspiel (Young 1994)

Stadt A (36.000 Einwohner) und Stadt B (12.000 Einwohner) planen je ein Wasserwerk.

Kosten Stadt A: **11 Mio. €**, d.h. $\frac{11.000.000}{36.000} \approx 306$ €/Einwohner

Kosten Stadt B: **7 Mio. €**, d.h. $\frac{7.000.000}{12.000} \approx 583$ €/Einwohner

Kosten eines gemeinsamen Wasserwerks für A und B: **15 Mio. €**

Problemstellung

Wie sollen die gesamten Kosten auf die Städte verteilt werden?

1. Jede Stadt trägt dieselbe Last.

Stadt A: 7,5 Mio Stadt B: 7,5 Mio

2. Jeder Einwohner trägt dieselbe Last

$\frac{15.000.000}{36.000+12.000} \approx 312 \text{ €/Einwohner}$
Stadt A: 11,25 Mio Stadt B: 3,75 Mio

3. Jede Stadt realisiert die gleiche Ersparnis.

Ersparnis $11+7-15 = 3$ Mio; 1,5 Mio je Stadt;
Stadt A: $11 - 1,5 = 9,5$ Mio. Stadt B: $7 - 1,5 = 5,5$ Mio.

4. Jeder Einwohner realisiert die gleiche Ersparnis.

Stadt A: $\frac{11.000.000}{36.000} - \frac{3.000.000}{48.000} \cdot 36.000 = 8,75$ Mio.

Stadt B: $\frac{7.000.000}{12.000} - \frac{3.000.000}{48.000} \cdot 12.000 = 6,25$ Mio.

(Nichtkooperative) Spieltheorie

Wie soll sich ein Spieler verhalten, um seinen Nutzen zu maximieren?

- ▶ strategieorientiert
- ▶ Aktionen, Strategien, Auszahlungen, Informationsmengen
- ▶ Lösung: Nash-Gleichgewicht

Kooperative Spieltheorie

Wie soll der **Nutzen** eines kooperativen Spiels auf die Spieler verteilt werden?

- ▶ auszahlungsorientiert
- ▶ Lösung: Auszahlung je Spieler, meist basierend auf Axiomen
- ▶ Es werden **keine** strategischen Spiele mit Absprachen zwischen Spielern betrachtet!

37 Motivation und Problemstellung

Kooperative Spiele

Der Shapley-Wert als ein Lösungskonzept

Zusammenfassung & Ausblick

Motivation und Problemstellung

Kooperative Spiele

Grundbegriffe

Gewichtete Abstimmungsspiele

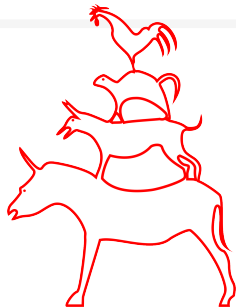
Der Shapley-Wert als ein Lösungskonzept

Punktwertige Lösungskonzepte

Shapley Axiome

Die Shapley-Formel

Anwendung der Shapley-Formel



- ▶ Betrachtet wird stets eine Menge von *Spielern* oder *Agenten*
 $N = \{1, 2, \dots, n\}$
- ▶ Spieler müssen keinesfalls gleichberechtigt sein.
- ▶ Eine **Koalition** K ist eine Teilmenge von N , $K \subseteq N$.

Welche/wie viele Koalitionen enthält die Menge $N = \{a, b, c\}$?

$$2^N = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$$

$$\text{Anz. Koalitionen: } |2^N| = |2^{\{a,b,c\}}| = 2^{|N|} = 2^3 = 8$$

- ▶ Die Menge aller Koalitionen ist 2^N (jeder Spieler kann *Mitglied* oder *nicht Mitglied* sein).
- ▶ N heiSSt *groSSe Koalition*.

- ▶ Ein *Koalitionsspiel* (N, v) besteht aus
- ▶ einer Menge N von $1, \dots, n$ Spielern sowie
- ▶ einer *Koalitionsfunktion* $v : 2^N \rightarrow \mathbb{R}$ mit $v(\emptyset) = 0$.

Die Funktion v ordnet jeder Koalition K den Nutzen (eine reelle Zahl $v(K)$) zu, der gemeinsame von den Mitgliedern in K geschaffen werden kann.

Der Wert $v(K)$ kann auf die Mitglieder in K aufgeteilt werden; wie dies erfolgt ist Gegenstand von Lösungskonzepten (Shapley, Kern)

- ▶ Für n Spieler lässt sich ein Abstimmungsspiel als Tupel

$$[q; g_1, \dots, g_n]$$

angeben. q ist die erforderliche Quote, g_i das Stimmgewicht von Spieler i .

- ▶ Erreicht eine Koalition die erforderliche Quote?

$$v(K) = \begin{cases} 1, & \text{falls } \sum_{i \in K} g_i \geq q \\ 0, & \text{falls } \sum_{i \in K} g_i < q \end{cases} \quad \begin{array}{l} K \text{ ist siegreich} \\ K \text{ ist unterlegen} \end{array}$$

Mehrheitsregel

$$\left[\frac{1}{2}; \frac{1}{n}, \dots, \frac{1}{n} \right]$$

- ▶ Sicherheitsrat: 5 ständige Mitglieder (USA, Russland, China, GB, F) 10 nicht-ständige Mitglieder
- ▶ Resolutionen müssen alle 5 ständigen Mitglieder zustimmen und mindestens 4 nicht-ständige.

[39; 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Wie lautet die Koalitionsfunktion des Sicherheitsrates?

Sei S die Menge der ständigen Mitglieder.

$$v(K) = \begin{cases} 1, & S \subseteq K \text{ und } |K| \geq 9 \\ 0, & \text{sonst.} \end{cases}$$

Motivation und Problemstellung

Kooperative Spiele

Grundbegriffe

Gewichtete Abstimmungsspiele

Der Shapley-Wert als ein Lösungskonzept

Punktwertige Lösungskonzepte

Shapley Axiome

Die Shapley-Formel

Anwendung der Shapley-Formel



- ▶ Eine **Lösung** für (N, v) ist ein **Auszahlungsvektor** $x = (x_1, \dots, x_n)$
- ▶ Jedem Spieler i aus N wird eine Auszahlung x_i zugeordnet (x wird auch Nutzenvektor genannt).
- ▶ Je nach Spiel: Nutzen x_i den Spieler i erhält bzw. Kosten, die i übernehmen muss

Bsp. Auszahlungsvektor

Wasserwerk-Spiel (Kosten Stadt A & Stadt B: 15 Mio. €)
 $(x_A, x_B) = (7.5, 7.5)$ oder $(14, 1)$ oder $(5, 20)$ oder $(5, 5)$

Lösungskonzepte:

- ▶ mengenwertig, d.h. eine Menge von Auszahlungsvektoren
- ▶ punktwertig, d.h. genau ein Auszahlungsvektor

Punktwertiges Lösungskonzept

Ein punktwertiges Lösungskonzept ψ ordnet jedem Koalitionsspiel genau einen Auszahlungsvektor zu.

1. Spieler $i \in N$ erhält alles:

$$\psi_j^{\text{alles für } i}(v) = v(N), \text{ falls } j = i, \text{ andernfalls } 0$$

2. Jeder Spieler $i \in N$ erhält z :

$$\psi_i^{\text{jeder erhält } z}(v) = z, i \in N$$

3. Egalitär:

$$\psi_i^{\text{egalitär}}(v) = \frac{v(N)}{n}, i \in N$$

4. Marginaler Beitrag zu N :

$$\psi_i^{\text{marginaler Beitrag zu } N}(v) = v(N) - v(N \setminus \{i\}), i \in N$$

Unter welchen Umständen gilt

1. ψ alles für $i = \psi$ egalitär i ,
2. ψ alles für $i = \psi$ jeder erhält z oder
3. ψ egalitär $= \psi$ jeder erhält z ?

zu 1 entweder $v(N) = 0$ oder $N = \{i\}$,

zu 2 sowohl $N = \{i\}$ als auch $v(N) = z$,

zu 3 falls $v(N)/n = z$.



Lloyd Shapley und Alvin Roth – Nobelpreis 2012 'for the theory of stable allocations and the practice of market design'

Shapley-Lösung

Die **Shapley-Lösung** φ ist dasjenige Lösungskonzept, das die folgenden Axiome erfüllt (Shapley 1953):

- ▶ Pareto-Axiom
- ▶ Symmetrie-Axiom
- ▶ Axiom über den unwesentlichen Spieler
- ▶ Additivitäts-Axiom

Operations Research

J. Pannek

Motivation und
Problemstellung

Kooperative Spiele

Der Shapley-Wert als
ein Lösungskonzept

Punktwertige
Lösungskonzepte

385 Shapley Axiome

Die Shapley-Formel

Anwendung der
Shapley-Formel

Zusammenfassung &
Ausblick

[Pareto-Axiom] Die Auszahlung $\psi_i(v)$, $\forall i \in N$ verteilt die maximal mögliche Nutzensumme:

$$\sum_{i \in N} \psi_i(v) = v(N).$$

Aufgabe: Welches der vier Konzepte $\psi_j^{\text{alles für } i}$, $\psi_i^{\text{jeder erhält } z}$, $\psi_i^{\text{egalitär}}$, $\psi_i^{\text{marginaler Beitrag zu } N}$ erfüllt das Pareto-Axiom?

ja: $\psi_j^{\text{alles für } i}$, $\psi_i^{\text{egalitär}}$

nein: $\psi_i^{\text{marginaler Beitrag zu } N}$, $\psi_i^{\text{jeder erhält } z}$, letzteres ja, falls $z = \frac{v(N)}{n}$.

Definition

Ein Spieler i , der zu jeder Koalition genau $v(i)$ beiträgt,

$$v(K \cup \{i\}) = v(K) + v(i), i \notin K, K \subseteq N,$$

heißt **unwesentlich**. Ein Spieler i , der zu keiner Koalition etwas beiträgt,

$$v(K \cup \{i\}) = v(K), K \subseteq N,$$

heißt **Nullspieler**.

Axiom (Unwesentlicher Spieler)

Jeder unwesentliche Spieler erhält genau den Wert seiner Einkerkoalition:

$$\psi_i(v) = v(\{i\}).$$

Axiom (Additivitäts-Axiom)

Wenn v und w Koalitionsfunktionen sind, dann soll für jeden Spieler $i \in N$ die Auszahlung bei $v + w$ gleich der Summe der Auszahlungen bei v und w sein:

$$\psi_i(v + w) = \psi_i(v) + \psi_i(w).$$

Die Gesamtauszahlung ist unabhängig davon, ob

- ▶ zwei Koalitionsfunktionen zuerst addiert werden und danach das Lösungskonzept angewendet wird, oder
- ▶ ob zuerst das Lösungskonzept auf jede der Koalitionsfunktionen einzeln angewendet wird und die Zahlungen danach addiert werden.

Axiom (Symmetrie-Axiom)

Die Auszahlungen hängen nicht von der Benennung der Spieler ab, sondern nur davon, was diese zu den Koalitionen beitragen.

Falls für jede Koalition K , die die Spieler i und j ($i \neq j$) nicht enthält,

$$v(K \cup \{i\}) = v(K \cup \{j\})$$

gilt, soll

$$\psi_i(v) = \psi_j(v)$$

gelten.

Jeder Spieler erhält den **Durchschnitt** seiner **marginalen Beiträge** zu **allen** möglichen Koalitionen.

Operations Research

J. Pannek

Motivation und
Problemstellung

Kooperative Spiele

Der Shapley-Wert als
ein Lösungskonzept

Punktwertige
Lösungskonzepte
Shapley Axiome

39 Die Shapley-Formel

Anwendung der
Shapley-Formel

Zusammenfassung &
Ausblick

- ▶ ρ bezeichnet eine **Reihenfolge** (ρ_1, \dots, ρ_n) der Spieler in N
- ▶ bei $N = \{1, 2, 3, 4\}$ z.B.
 $\rho^1 = (3, 1, 2, 4), \rho^2 = (2, 4, 3, 1), \rho^3 = (1, 3, 2, 4), \dots$
- ▶ $K_i(\rho)$ bezeichnet die Koalition bestehend aus den Spielern $\rho_1, \rho_2, \dots, \rho_i$ (also ohne die Spieler $\rho_{i+1}, \dots, \rho_n$).

Beispiele

Reihenfolge $\rho = (3, 1, 4, 2)$ und Spieler $i = 1$:

$$K_1((3, 1, 4, 2)) = \{1, 3\}$$

Reihenfolge $\rho = (2, 3, 1, 4)$ und Spieler $i = 4$:

$$K_4((2, 3, 1, 4)) = \{2, 3, 1, 4\}$$

Reihenfolge ρ^3 und Spieler $i = 3$: $K_3(\rho^3) = K_3((1, 3, 2, 4)) = \{1, 3\}$

- ▶ Welchen Wert hat eine Koalition K mit i und welchen ohne i ?
- ▶ Marginaler Beitrag eines Spielers i zur Koalition K :

$$MB_i^K(v) = v(K \cup \{i\}) - v(K \setminus \{i\}).$$

- ▶ Alternative Schreibweise bei gegebener Beitrittsreihenfolge ρ :

$$MB_i^\rho(v) = v(K_i(\rho) \cup \{i\}) - v(K_i(\rho) \setminus \{i\}).$$

Bestimmen Sie die marginalen Beiträge $MB_B^{(A,B)}$ und $MB_B^{(B,A)}$ im Wasserwerk-Spiel

($c(A, B) = 15$, $c(A) = 11$, $c(B) = 7$, $c(\{\}) = 0$).

$$MB_B^{(A,B)} = 15 - 11 = 4$$

$$MB_B^{(B,A)} = 7 - 0 = 7$$

- ▶ Addiere für alle möglichen Beitrittsreihenfolgen (bzw. für alle Koalitionen) die marginalen Beiträge des Spielers i und dividiere durch die Anzahl möglicher Beitrittsreihenfolgen.
- ▶ Für $n = |N|$ gibt es $n! = n \cdot (n - 1) \cdot \dots \cdot 1$ Beitrittsreihenfolgen.

Beispiel: Shapley-Zahlung $\varphi_B(c)$ für Stadt B im Wasserwerk-Spiel $N = \{A, B\}$,
 $c(A, B) = 15$, $c(A) = 11$, $c(B) = 7$, $c(\{\}) = 0$

1. Reihenfolgen: (A,B), (B,A)
2. Summe marginaler Beiträge Spieler B für alle Reihenfolgen:

$$MB_B = MB_B^{(A,B)} + MB_B^{(B,A)} = 4 + 7 = 11$$

3. Shapley-Zahlung für Spieler B: $\varphi_B(c) = \frac{1}{n!} \cdot MB_B = \frac{1}{2} \cdot 11$

Die Shapley-Formel lautet

$$\begin{aligned}\psi_i^{\text{Shapley}}(v) &= \varphi_i(v) \\ \varphi_i(v) &= \frac{1}{n!} \sum_{\rho \in RF} MB_i^\rho(v) \\ &= \frac{1}{n!} \sum_{\rho \in RF} [v(K_i(\rho)) - v(K_i(\rho) \setminus \{i\})].\end{aligned}$$

Die Shapley-Lösung für ein Koalitionsspiel (N, v) lautet

$$\varphi(v) = (\varphi_1(v), \dots, \varphi_n(v)).$$

Anwendung Shapley-Formel entweder direkt auf
Kostenaufteilungsfunktion c oder auf Kostenersparnisfunktion v .

$$c(\{A\}) = 11$$

$$c(\{B\}) = 7$$

$$c(\{A, B\}) = 15$$

Wie lauten die Shapley-Zahlungen φ_A, φ_B ?

Reihenfolgen ρ	MB_A^ρ	MB_B^ρ
(A,B)	11	4
(B,A)	8	7
Summe	19	11
Shapley-Zahlungen φ_i	$\frac{19}{2}$	$\frac{11}{2}$

Drei Bergbaukonzerne fördern in Brasilien Eisenerz. Zum Abtransport des Erzes aus den Minen hin zu mehreren Seehäfen unterhalten die drei Bergbaukonzerne ein gemeinsames Eisenbahnnetz. Gegeben ist die nachfolgende Kostenfunktion, welche jeder Teilkoalition die jährlich zu tragenden Gesamtkosten (in Geldeinheiten GE) des Eisenbahnnetzes zuordnet.

$$c(1) = 400, \quad c(2) = 500, \quad c(3) = 500$$

$$c(1, 2) = 750, \quad c(1, 3) = 750, \quad c(2, 3) = 900$$

$$c(1, 2, 3) = 1050$$

Die drei Bergbaukonzerne suchen nach einer geeigneten Aufteilung der jährlichen Gesamtkosten in Höhe von 1050 GE.

Welche Kosten müssen die drei Bergbaukonzerne (Spieler 1, 2 und 3) jeweils gemäss der Shapley-Formel tragen?

Shapley-Formel für ein Eisenbahnkoalitionsspiel 2/2

Reihenfolgen ρ	MB_1^ρ	MB_2^ρ	MB_3^ρ
(1,2,3)	400-0=400	350	300
(1,3,2)	400-0=400	300	350
(2,1,3)	750-500=250	500	300
(2,3,1)	1050-900=150	500	400
(3,1,2)	750-500=250	300	500
(3,2,1)	1050-900=150	400	500
Summe	1600	2350	2350
Shapley-Zahlung φ_i	$\frac{1600}{6} \sim 267$	$\frac{2350}{6} \sim 392$	$\frac{2350}{6} \sim 392$

Vergleichen Sie die Shapley-Auszahlung der beiden Abstimmungsspiele $[70; 50, 25, 25]$ und $[70; 55, 35, 10]$ für Spieler 1

$$a) \varphi_1 = \frac{4}{6}$$

$$b) \varphi_1 = \frac{3}{6}$$

Trotz Erhöhung des Stimmgewichts von Spieler 1 sinkt dessen Shapley-Auszahlung, weil sein marginaler Beitrag zu allen Koalitionen (insb. mit Spieler 3) sinkt.

Fragen zur Wiederholung:

- ▶ Was bedeutet der **Shapley Wert** als punktwertiges Lösungskonzept für Koalitionsspiele?
- ▶ Was ist ein **Kooperationsspiel** und eine **Kooperationsfunktion**?

Literatur

Wiese 2004 (klick mich), Kapitel C1-C4, F1 bis F6