0. Einleitung

0.1. Der Begriff Operations Research / Decision Support

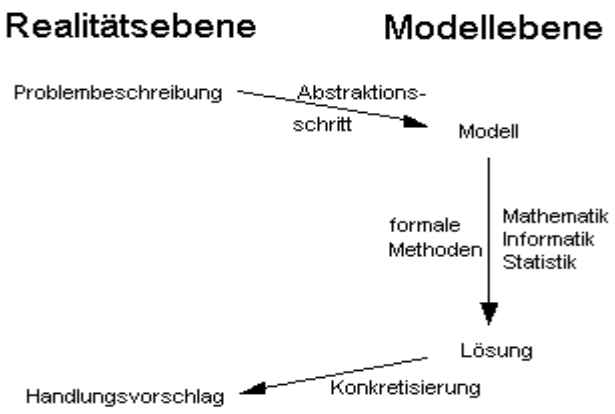
Es werden Methoden zur **Entscheidungsunterstützung** (Decision Support, DS) vorgestellt. Durch Problemanalyse, daraus formulierte mathematische (formale) Modelle und/oder Computerverfahren sollen **optimale** Entscheidungen vorbereitet und untertstützt werden. Einsatzgebiete gibt es in Wirtschaftsplanung, betrieblicher Planung, technischen Bereichen, auch in der Informatik u.a.m.

Historisch aus Verfahren entstanden, die bis in die Jahre vor bzw. nach dem 2. Weltkrieg zurückreichen. Unter dem Schlagwort Operations Research (OR, Unternehmensforschung) bekannt geworden. In letzter Zeit Verschiebung von den klassischen Verfahren des OR zu "entscheidungsunterstützenden Methoden":

- Stärkere Einbindung von Methoden aus der Informatik (AI-Ansätze in den sog. Decision Support Systems)
- Zurücktreten des reinen Optimierungsaspekts ("Welche Lösung ist die beste?") gegenüber sogenannten Heuristiken ("Wie komme ich zu einer einigermaßen guten Lösung?"). Ausgelöst durch den enormen Komplexitätszuwachs moderner betrieblicher Strukturen, aber auch technischer bzw. informationstechnischer Systeme (z.B. Internet).
- Stärkere Betonung der Interaktion zwischen Entscheidungsträger und entscheidungsunterstützendem System. Die Erwartungshaltung, daß ein System nur die Problembeschreibung braucht, um daraus fix und fertige Lösungen ableiten zu können, ist überholt.

Trotzdem bilden die "Kernverfahren" des OR noch immer das Skelett des Gebietes Decision Support (bes. die Optimierungsverfahren).

Das Grundschema der DS-Methoden ist das der Modellbildung:



Die **Methoden** des DS behandeln den **Weg vom Modell zur Lösung**.

0.2. Gliederung anwendungs-/methodenorientiert

0.2.1. Anwendungsspezifische Gliederung:

DS Verfahren werden oft nicht nach methodischen Gesichtspunkten, sondern nach Anwendungen gegliedert.

1. Banken, Finanzierung, Versicherungen
    1. Finanzplanung
    2. Investition
    3. Versicherungsmathematik
2. Produktion
    1. Lagerhaltung
    2. Maschinenbelegung
    3. Transport (und Verkehr)
    4. Flexible Manufacturing System (FMS)
3. Logistik
4. Zuverlässigkeit
5. Marketing
6. Makroökonomie
7. Angewandte Ökonomie
8. Schnittstellen zur Informatik
9. Energieversorgung
10. Umweltfragen
11. Lagerhaltung
12. Gesundheitswesen
13. Personaleinsatz
14. Projektmanagement
15. Risikomanagement

Ein und dasselbe methodische Verfahren kommt in verschiedenen Anwendungsbereichen vor, z.B. Lineare Optimierung in Transport, Finanzierung, Produktion, usw.

0.2.2. Methodisch

können wir die Verfahren folgendermaßen gliedern:

1. Optimierung
2. Graphen und Netzpläne
3. Diskrete Ereignissysteme und Simulation
4. Entscheidungsunterstützende Systeme und Expertensysteme

ad 1. Optimierung

Wir unterscheiden zwischen **Zielfunktion** (Maximum oder Minimum) und **Nebenbedingungen** (schränken die möglichen Handlungsalternativen ein). Je nachdem, wie man die Frage stellt, sind Zielfunktion und Nebenbedingungen vertauschbar.

Beispiel 0.1: *Gewinn versus Umweltschutz*

Variante 1: Gewinnmaximierung unter Emissions-Nebenbedingungen
Variante 2: Emissionsminimierung unter Kosten-Nebenbedingungen
Variante 3: Gewinnmaximierung **und** Emissionsminimierung. Geht so direkt nicht. Sinnvoll nur machbar durch Gewichtung der verschiedenen Ziele, z.B.

Zielfunktion = α * Gewinn - β * Schadstoffemission. Das ist eine sogenannte "Mehrzieloptimierung".

Oft werden die Nebenbedingungen durch Einführung von Strafkosten ("penalty functions") in die Zielfunktion eingebaut.

Mathematische Notation:

$$\begin{cases} F(x) \rightarrow max! \qquad oder \qquad F(x) \rightarrow min! \\ x \in S \qquad S\ ist\ die\ Menge\ der\ zulässigen\ Alternativen \end{cases}$$

Je nach Beschaffenheit von **S** lassen sich folgende Unterfälle unterscheiden:

a) **S** ist **endliche** Menge mit kombinatorischer Struktur
Kombinatorische Optimierung (z.B. Stundenplanerstellung)

b) **S** ist eine Menge von **ganzzahligen Vektoren**
Ganzzahlige Optimierung
S ist - zumindest im Prinzip - unendlich.
Elemente von **S** (Handlungsalternativen) sind von der Form $(x_1, ..., x_n)^T$ mit x_i ganzzahlig.
Beispiele:

- Personaleinsatzpläne
- Maschineneinsatzpläne
- Transportmitteleinsatzpläne: z.B. Güter sollen von Produzenten zu Konsumenten transportiert werden. Wieviele LKWs sollen pro Strecke eingesetzt werden?
- Ganzzahlige Mischungsprobleme: z.B. Kaffee soll in Qualität gleich bleiben. Wieviele Container von welchem Anbaugebiet? (jedes Jahr zu entscheiden)

a) und b) sind nicht immer klar voneinander abgrenzbar.

c) **S** ist Menge von **reellwertigen Vektoren**
Kontinuierliche Optimierung
Elemente von **S** (Handlungsalternativen) sind von der Form $(x_1, ..., x_n)^T$ jetzt aber mit x_i reellwertig.
Beispiele:

- Kontinuierliche Mischungsprobleme: z.B. aus Rohölen Benzin mischen (NB Dichte, Viskosität, Oktananzahl; ZF Kosten)
- Rohstoffeinsatzpläne: z.B. Energieversorgung
- Portfolio-Split (Geld ist "fast" kontinuierlich)
- Kapazitätsplanung: z.B. Bau von Lagerräumen

Weiter untergliederbar:
c1) Lineare Optimierung

ZF ist linear und die NB werden durch lineare (Un)gleichungen beschrieben.

Beispiel 0.2.: *3 Arten von Schweinefutter werden gemischt*

Preise	p_1	p_2	p_3
Eiweiß	e_1	e_2	e_3
Kalorien	k_1	k_2	k_3
Menge	x_1	x_2	x_3

ZF: $p_1x_1 + p_2x_2 + p_3x_3 \rightarrow$ min!

NB: $e_1x_1 + e_2x_2 + e_3x_3 \geq e$

$k_1x_1 + k_2x_2 + k_3x_3 \geq k$

Beispiel 0.3.:*Inhalt einer Vorlesung*

m mathematische Theorie	Gewicht 1
v Verfahren	Gewicht 5
b Beispiele	Gewicht 7

ZF: $1m + 5v + 7b \rightarrow$ max!

NB: $m + v + b = 1$

$m \geq 0.1$

$m + v \geq b$

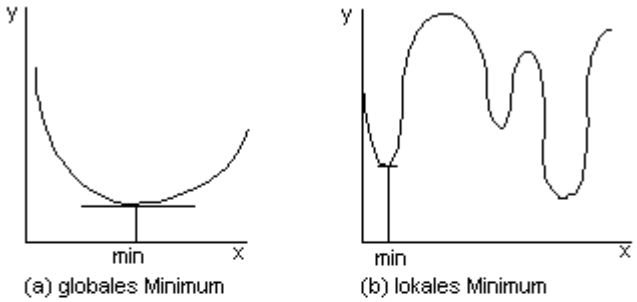
$b \geq 0.3\ (m + v)$

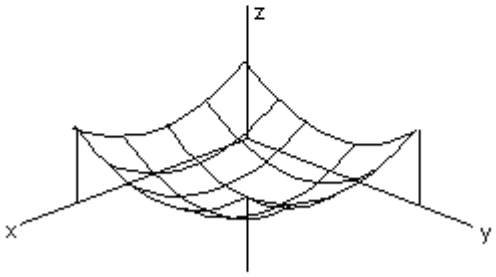
c2) Konvexe Optimierung

ZF ist konvex und die NB werden durch konvexe (Un)gleichungen beschrieben.

c3) Nichtlineare Optimierung

ZF oder NB nicht linear bzw. konvex.





(c) globales Minimum (3-dimensional)

d) **Kette von Entscheidungsschritten**: **S** ist so strukturiert, daß es in mehrere Entscheidungen zeitlich hintereinander zerfällt.

Dynamische Optimierung

Die Elemente von **S** heissen **Strategien**.

Beispiele:

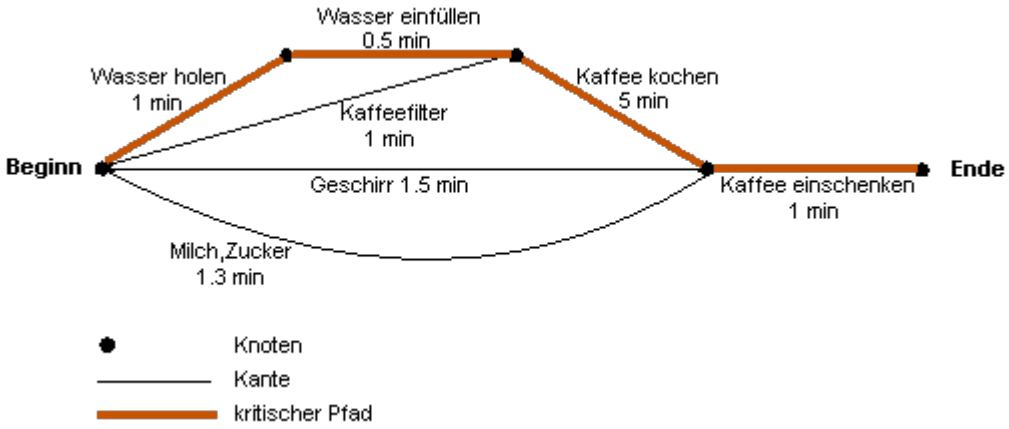
- Kraftwerkseinsatzplan: Jede Stunde fällt Entscheidung, ob Kraftwerke hinzu- oder abgeschaltet werden (Parameter: Strombedarf, GAU-Gefahr, ...)
- Aktienmarkt: Entscheidungen über Kauf und Verkauf von Aktien.

Falls Zeitablauf nicht diskret, sondern kontinuierlich behandelt wird: Kontrolltheorie

ad 2. Graphen und Netzpläne

Manche Handlungsabläufe oder Entscheidungen lassen sich gut durch Graphen repräsentieren. Bei großen Industrieprojekten kommen mehrere 100 oder sogar mehrere 1000 Knoten vor.

Beispiel 0.4. *Projekt "Kaffee kochen"*



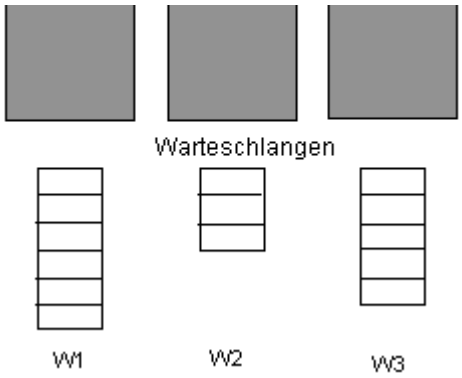
Die bekannteste Netzplantechnik ist wohl CPM (critical path method): Hierbei wird der kritische Weg, d.h. der längste Weg vom Projektbeginn zum Projektende gesucht. Seine Länge bestimmt die (minimale) Gesamtdauer des Unternehmens. Verzögerungen entlang des kritischen Weges führen unweigerlich zu Verzögerung des gesamten Projekts.

Die Optimierung graphentheoretischer Entscheidungen ist Teil der kombinatorischen Optimierung (endliche Anzahl von Handlungsalternativen).

ad 3. Diskrete Ereignissysteme und Simulation

Produktions- und Transportsysteme können als diskrete Ereignissysteme modelliert und analysiert werden.
Beispiel: Finden optimaler Maschinenzahlen und Verteilungs (Scheduling) - Strategien in Fertigungssystemen.

Beispiel 0.5. *Servicezentren, Kunden in Warteschlange*



"diskret": Kunde kommt oder er kommt nicht.

Analyse auf zwei Arten:

- Mathematische Theorie: Warteschlangentheorie
- Simulationsmethoden: Ereignissimulation

ad 4. Entscheidungsunterstützende Systeme und Expertensysteme

Schwergewicht liegt auf der Schnittstelle Mensch/Maschine. Optimierung von Entscheidungen in einem interaktiven Prozeß.

a) Entscheidungsunterstützende Systeme (EUS, DDS=Decision Support Systems)
Interaktive Benutzerschnittstelle + klassische OR Verfahren (Optimierung, Graphentheorie, Simulation)
Der Benutzer hat gewisse Wahlmöglichkeiten, bekommt keine fertige Lösung. Er kann mit dem System "spielen".
Beispiel: Produktionsprozeß evtl. graphisch darstellen.
Modelle sind parametrisierbar (Anzahl/Leistungsfähigkeit der Maschinen, Ankunftsprozeß der Werkstücke, Zuteilungsstrategie (Scheduling)
Ergebnisse sind etwa Verweildauer, Gesamtdurchsatz, (mittlere) Warteschlangenlängen.
Der OR Kern wäre hier wohl ein Simulationsverfahren.

b) Expertensysteme
Benutzerschnittstelle + Regelsysteme der formalen Logik oder künstlicher Intelligenz (z.B. neuronale Netze)
evtl. erweitert um wahrscheinlichkeitstheoretisches (stochastisches) Modell.
Grenzbereich zur Informatik.
Beispiel: Diagnose von Krankheiten. "Facharztwissen" wird in Form einer Regelbasis gespeichert.
Benutzer gibt Symptome, Befunde, etc. ein und erhält eine (oder mehrere) mögliche Diagnose(n).
Systeme dieser Art werden in letzter Zeit zunehmend auch im DS-Bereich eingesetzt (z.B. Maschinenbelegungsplanung)

0.3. Verhältnis Statistik - Operations Research

Übergänge fließend.

- Falls Datenproblematik vorherrschend → statistische Methoden (z.B. Absatzprognosen)
- Datenerhebung statistisch unproblematisch → OR-Methoden

Stochstische Modelle spielen in OR vielfach eine Rolle. Z.B. immer dann wenn eine Ungewißheit über Entscheidungsgrundlagen vorhanden ist.
Beispiel: Warteschlangenmodelle. Der Zustrom ist zufälliger Natur.
Statt "Optimierung" dann " Stochastische Optimierung".

0.4. Decision Support und Information

a) **Entscheidungen unter Sicherheit**
In diesen Fällen ist keine Ungewißheit vorhanden. Man verwendet die Methoden der klassischen Optimierung.

b) **Entscheidung unter Risiko**
Quantifizierbare Ungewißheit vorhanden. Stochastische Modelle werden verwendet. Stochastische Optimierung.
Erwartungswert und Varianz werden als Entscheidungskriterium herangezogen.

c) **Entscheidung unter Unsicherheit**
Nicht quantifizierbare Ungewißheit vorhanden. Spieltheoretische Modelle werden verwendet. Minimax-Regeln werden herangezogen.

Beispiel 0.6. *Entscheidung unter Sicherheit, Risiko, Unsicherheit*
ad a) Angenommen man kann zwischen zwei Alternativen A und B wählen. A bringt einen Gewinn von 10.000, B einen Gewinn von 4.000.
Die Entscheidung ist offensichtlich: wähle A.
ad b) Z.B. Einfluss von Schön- und Schlechtwetter auf Gewinn:

	Schönwetter	Schlechtwetter
Alternative A	10.000	3.000
Alternative B	4.000	5.000

Falls Schön- und Schlechtwetter gleich wahrscheinlich sind, berechnen sich die Erwartungswerte wie folgt:
E(Gewinn | A) = 0,5 * 10.000 + 0,5 * 3.000 = 6.500
E(Gewinn | B) = 0,5 * 4.000 + 0,5 * 5.000 = 4.500
Man wählt Alternative A, da sie den höheren Erwartungswert hat (allerdings hat B den höheren sicheren Gewinn)
ad c) Z.B. ein Konkurrent, dessen Erwartungswerte man nicht kennt. Konkurrent kann sich für Alternative C oder D entscheiden.
Entsprechend ändern sich die Gewinne.

	Alternative C	Alternative D
Alternative A	10.000	3.000
Alternative B	4.000	5.000

Man muß davon ausgehen, daß der Konkurrent seine Entscheidungen so trifft, daß sein Gewinn maximiert wird und damit der eigene minimiert.
Die Entscheidung muß also so getroffen werden, daß dieses Minimum maximiert wird (konservatives, vorsichtiges Verhalten).
Im obigen Fall wäre daher die Entscheidung B zu treffen (Maximum der Minima).

1. Lineare Optimierung

1.1. Anwendungsbeispiele und allgemeine Problemformulierung

Beispiel 1.1. *Mischungsproblem*: Eine Tierfarm kauft 3 verschiedene Kornsorten S_1, S_2, S_3 um daraus Futtermittel zu mischen.

Kornsorten	**S_1**	**S_2**	**S_3**	**Mindestbedarf**
Nährstoff A	2	3	7	1250
Nährstoff B	1	1	0	250
Nährstoff C	5	3	0	900
Nährstoff D	0.6	0.25	1	232.5
Kosten/Einheit	41	35	96	

Wie sollen die Kornsorten gemischt werden, um den Nährstoffbedarf möglichst kostengünstig zu decken?

Sei x_i die gekaufte Menge der Sorte S_i (i=1,2,3)

Mathematisch wird das Problem folgendermaßen formuliert:

Zielfunktion:	$41\,x_1 + 35x_2 + 96x_3$	$\rightarrow$	min!
Nebenbedingungen:	$2\,x_1 + 3\,x_2 + 7\,x_3$	$\geq$	1250
	$x_1 + x_2$	$\geq$	250
	$5\,x_1 + 3\,x_2$	$\geq$	900
	$0.6\,x_1 + 0.25\,x_2 + x_3$	$\geq$	232.5

Vorzeichenbedingungen: $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$

Beispiel 1.2. *Transportproblem*: 3 Kiesgruben beliefern 4 Baustellen.
Tagesaufkommen der Kiesgruben: K1 = 12t, K2 = 3t, K3 = 11t
Bedarf der Baustellen: B1 = 5t, B2 = 4t, B3 = 7t, B4 = 10t

Transportkosten pro Tonne von Kiesgrube zu Baustelle (Transportmatrix):

	B1	B2	B3	B4
K1	8	15	10	17
K2	25	13	7	14
K3	4	10	3	8

Wieviel soll von den einzelnen Kiesgruben zu den einzelnen Baustellen geliefert werden, sodaß die Gesamttransportkosten minimal sind?

Sei x_{ij} die Liefermenge von Ki nach Bj.

Mathematische Formulierung:

ZF $8x_{11} + 15x_{12} + 10x_{13} + 17\,x_{14} +$
$25x_{21} + 13x_{22} + 7x_{23} + 14\,x_{24} +$
$4x_{31} + 10x_{32} + 3x_{33} + 8\,x_{34} \quad \rightarrow$ min!

NB Produktion der Kiesgruben

$x_{11} + x_{12} + x_{13} + x_{14} = 12$

$x_{21} + x_{22} + x_{23} + x_{24} = 3$

$x_{31} + x_{32} + x_{33} + x_{34} = 11$

Bedarf der Baustellen

$x_{11} + x_{21} + x_{31} = 5$

$x_{12} + x_{22} + x_{32} = 4$

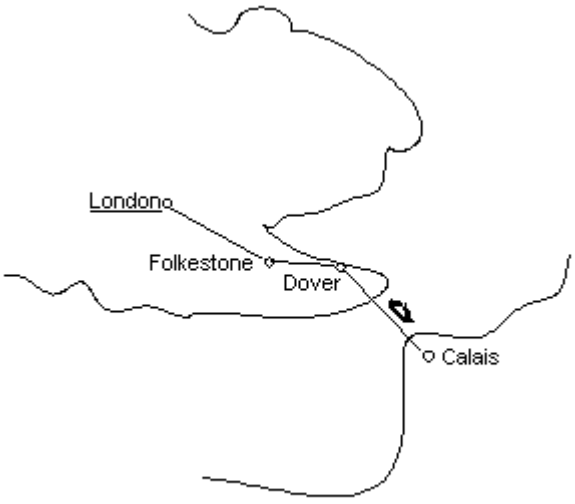
$x_{13} + x_{23} + x_{33} = 7$

$x_{14} + x_{24} + x_{34} = 10$

Nichtnegativitätsbedingungen: $x_{ij} \geq 0$ (i=1,...,3, j=1,...,4)

Hier sind die Nebenbedingungen nicht mehr unabhängig! Eine Nebenbedingung ist redundant, man kann daher etwa die letzte NB weglassen.

Beispiel 1.3. *Spieltheorie*:
Sherlock Holmes wird verfolgt. Sein Feind, Dr. Moriarty, sitzt im selben Zug in einem anderen Waggon. Es gibt noch 2 Stationen: Folkestone (F), Dover (D). Steigen beide bei derselben Station aus, wird Holmes getötet. Steigt Holmes bei D aus und sein Feind bei F, so ist Holmes gerettet. Steigt er bei F aus und sein Feind bei D, so hat Holmes 50% Überlebenschance.



"Auszahlungsmatrix":

	Moriarty

Holmes		F	D
	F	0	1/2
	D	1	0

Moriarty ist äußerst intelligent. Empfiehlt dieSpieltheorie für Holmes die Alternative F, so "weiß " das Moriarty und wird sich für F entscheiden. Empfiehlt die Spieltheorie hingegen D, so wird auch Moriarty
so wählen. Bei Minimax-Prinzip ist die Auszahlung für Holmes = 0.

"**Gemischte Strategie**": mit Wahrscheinlichkeit p bei F aussteigen, mit Wahrscheinlichkeit 1-p bei D.

Welche Strategie p auch immer als optimale Strategie herauskommt, man muß davon ausgehen, daß auch der Feind sie berechnen kann.

Sei w die Überlebenswahrscheinlichkeit für Holmes.

Mathematische Formulierung:

ZF:	w	$\rightarrow$ max!	
NB:	w	$\leq$ 1-p = 0.p + 1.(1-p)	"Moriarty steigt in F aus"
	w	$\leq$ 0.5p = 0.5p + 0.(1-p)	"Moriarty steigt in D aus"

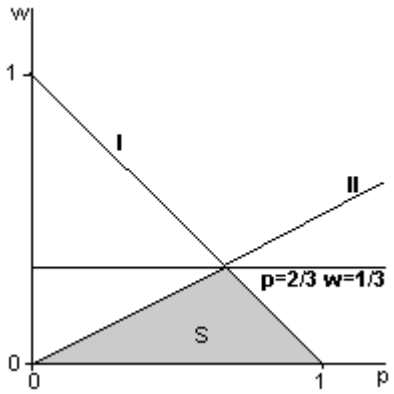
Nichtnegativität: w, p $\geq$ 0

Etwas umformuliert:

ZF:	w	$\rightarrow$	max!
NB:	w + p	$\leq$	1
	w - 0.5p	$\leq$	0

Nichtnegativität: w, p $\geq$ 0 (außerdem p$\leq$1)

Grafische Lösung:



1.2. Allgemein formuliertes Lineares Programm

Lineare Optimierungsprobleme werden als Lineares Programm (LP) bezeichnet.

c_1x_1	+...+	c_nx_n	$\rightarrow$	max!	Zielfunktion
$a_{11}x_1$	+...+	$a_{1n}x_n$	=	b_1	
...	+...+	...			Gleichungen
$a_{k1}x_1$	+...+	$a_{kn}x_n$	=	b_k	
$a_{k+1,1}x_1$	+...+	$a_{k+1,n}x_n$	$\leq$	b_{k+1}	
...	+...+	...			Ungleichungen
$a_{m1}x_1$	+...+	$a_{mn}x_n$	$\leq$	b_m	
		$x_1 \geq 0$			
		...			Vorzeichenbeschränkung ($s \leq n$)
		$x_s \geq 0$			

In Vektorschreibweise:

Entscheidungsvariable:	$\mathbf{x} = (x_1, ..., x_n)^T$
Zielfunktionskoeffizienten:	$\mathbf{c} = (c_1, ..., c_n)^T$
Gleichheits-Nebenbedingungen:	$\mathbf{A_1} = (a_{ij})$ $1 \leq i \leq k,\ 1 \leq j \leq n$
Ungleichheits-Nebenbedingungen:	$\mathbf{A_2} = (a_{ij})$ $k+1 \leq i \leq m,\ 1 \leq j \leq n$
"rechte Seite":	$\mathbf{b} = (\mathbf{b_1}, \mathbf{b_2})^T$ $\mathbf{b_1} = (b_1,...,b_k)^T$ $\mathbf{b_2} = (b_{k+1},...,b_m)^T$

n: Anzahl der Entscheidungsvariablen
m: Anzahl der Nebenbedingungen

Lineares Programm wird dann so geschrieben:

ZF:	$\mathbf{c}^T\mathbf{x}$	$\rightarrow$	max!
NB:	$\mathbf{A_1}\,\mathbf{x}$	=	$\mathbf{b_1}$
	$\mathbf{A_2}\,\mathbf{x}$	$\leq$	$\mathbf{b_2}$
Nichtnegativität	x_j	$\geq$	$0,\ 1 \leq j \leq s$

1.3. Transformation Linearer Optimierungsprobleme auf LP's in allgemeiner Form

- Maximieren statt Minimieren der Zielfunktion: Ersetze $\mathbf{c}^T$ durch $-\mathbf{c}^T$, also $-\mathbf{c}^T\mathbf{x} \rightarrow$ max!
- Ungleichungen der Form $\geq$: Multipliziere die NB mit (-1), es entspricht also $\mathbf{A_2}\,\mathbf{x} \geq \mathbf{b_2}$ der Nebenbedingung $-\mathbf{A_2}\,\mathbf{x} \leq -\mathbf{b_2}$

  Negative Vorzeichenbeschränkungen ($x_j \leq 0$): Variablentransformation. Nimm statt der Variablen x_j die Variable $y_j = -x_j$, dann gilt $y_j \geq 0$.

1.4. Standardform

Wenn keine Gleichheitsnebenbedingungen vorhanden sind und alle x_j vorzeichenbeschränkt sind, dann spricht man von der **Standardform** des LP.

ZF:	$\mathbf{c}^T\mathbf{x} \rightarrow$	max!
NB:	$\mathbf{A}\,\mathbf{x} \leq$	$\mathbf{b}$
Nichtnegativität	$\mathbf{x} \geq$	$\mathbf{0}$

A hat Rang n (n = Anzahl der Variablen)

1.5. Transformation der allgemeinen Form zur Standardform

Ein LP in allgemeiner Form kann stets auf ein LP in Standardform gebracht werden.

1. Gleichheitsnebenbedingungen zu Ungleichheitsnebenbedingungen transformieren: Aus einer Gleichheitsnebenbedingung werden 2 Ungleichheitsnebenbedingungen.
   $a_{i1}x_1 + ... + a_{in}x_n = b_i$ wird zu $a_{i1}x_1 + ... + a_{in}x_n \leq b_i$ und $-a_{i1}x_1 - ... - a_{in}x_n \leq -b_i$
2. Fehlende Vorzeichenbeschränkung: Sei x_j nicht vorzeichenbeschränkt.Dann kann man x_j folgendermaßen in pos. (x_j^+) und neg. (x_j^-) Anteil zerlegen: $x_j = x_j^+ - x_j^-$ (einer dieser beiden Anteile ist gleich 0).
   $x_j^+ = x_j$, falls $x_j \geq 0$ und $x_j^+ = 0$ sonst.
   $x_j^- = 0$, falls $x_j \geq 0$ und $x_j^- = -x_j$ sonst.
   Somit ist sowohl $x_j^+ \geq 0$, als auch $x_j^- \geq 0$.
3. Vollen Zeilenrang herstellen: redundante Zeilen werden gestrichen.

1.6. Transformation von Ungleichungen zu Gleichungen

Es werden **Schlupfvariable** eingeführt.
Statt $a_{i1}x_1 + ... + a_{in}x_n \leq b_i$ schreibt man nun $a_{i1}x_1 + ... + a_{in}x_n + u_i = b_i$, wobei $u_i \geq 0$ eine zusätzliche "Schlupf"variable ist.
Man erhält die "**erweiterte Standardform**":

ZF:	$\mathbf{c}^T\mathbf{x}$	$\rightarrow$	max!
NB:	$\mathbf{B}\,\mathbf{x}$	=	$\mathbf{b}$
Nichtnegativität	$\mathbf{x}$	$\geq$	$\mathbf{0}$

Hier enthält nun der Variablenvektor **x** zusätzlich auch Schlupfvariablen, die Matrix **B** entspricht der Matrix **A**, erweitert um die nötigen Spalten. Die Dimension des Problems hat sich erhöht!
Oft wird auch noch $\mathbf{b} \geq \mathbf{0}$ verlangt. Dies erhält man durch Multiplizieren mit (-1) der Gleichungen, wo dies notwendig ist.

Beispiel 1.4. Fortsetzung des Mischungsproblems 1.1.

In **Standardform** bringen:

$-41x_1$	$-35x_2$	$-96x_3$	$\rightarrow$	max!
$-2x_1$	$-3x_2$	$-7x_3$	$\leq$	-1250
$-x_1$	$-x_2$		$\leq$	-250
$-5x_1$	$-3x_2$		$\leq$	-900
$-0.6x_1$	$-0.25x_2$	$-x_3$	$\leq$	-232.5

$x_1, x_2, x_3 \geq 0$

In **Vektorschreibweise**:

n=3
$\mathbf{x} = (x_1, x_2, x_3)^T$
$\mathbf{c} = (-41, -35, -96)^T$

$$A = \begin{bmatrix} -2 & -3 & -7 \\ -1 & -1 & 0 \\ -5 & -3 & 0 \\ -0,6 & -0,25 & -1 \end{bmatrix} \qquad b = \begin{bmatrix} -1250 \\ -250 \\ -900 \\ -232,5 \end{bmatrix}$$

ZF:	$\mathbf{c}^T\mathbf{x}$	$\rightarrow$	max!
NB:	$\mathbf{A}\,\mathbf{x}$	$\leq$	$\mathbf{b}$
Nichtnegativität	$\mathbf{x}$	$\geq$	$\mathbf{0}$

Erweiterte Standardform: Einführung von Schlupfvariablen

$$
\begin{array}{llllll}
-2x_1 - 3x_2 - 7x_3 & +u_1 & & & & = -1250 \\
-x_1 - x_2 & & +u_2 & & & = -250 \\
-5x_1 - x_2 & & & +u_3 & & = -900 \\
-0,6x_1 - 0,25x_2 - x_3 & & & & +u_4 & = -232,5
\end{array}
$$

Transformiertes Modell:

$$\widetilde{n} = 7$$
$$\widetilde{x} = (x1, x2, x3, u1, u2, u3, u4)$$
$$\widetilde{c} = (-41\ -35\ -96,\ 0,\ 0,\ 0, 0)$$

$$\widetilde{A} = \begin{pmatrix} -2 & -3 & -7 & 1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 1 & 0 & 0 \\ -5 & -3 & 0 & 0 & 0 & 1 & 0 \\ -0.6 & -0.25 & -1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\widetilde{b} = b$$

Beispiel 1.5. Fortsetzung des Transportproblems 1.2.

$$n = 12$$
$$x = (x_{11}, ..., x_{14}, x_{21}, ..., x_{24}, x_{31}, ..., x_{34})^T$$
$$c = (8, 15, 10, 17, 25, 13, 7, 14, 4, 10, 3, 8)^T$$

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

$$b^T = (12, 3, 11, 5, 4, 7, 10)$$
$$x_{ij} \geq 0, 1 \leq i \leq 3, 1 \leq j \leq 4$$

Die Matrix **A** hat Rang 6. Eine Matrix mit vollem Rang erhält man etwa durch Streichen der letzten Zeile.

Beispiel 1.6. Fortsetzung Sherlock Holmes 1.3.

In **Standardform** bringen:

w	$\rightarrow$	max!
w + p	$\leq$	1
w - 0.5p	$\leq$	0
w, p	$\geq$	0

In **Vektorschreibweise**:

$$
\begin{aligned}
&n = 2 \\
&x = (w, p)^T \\
&c = (1, 0)^T \\
&A = \begin{pmatrix} 1 & 1 \\ 1 & -\frac{1}{2} \end{pmatrix} \\
&b = (1, 0)^T
\end{aligned}
$$

Erweiterte Standardform:

w	$\rightarrow$	max!
w + p + u	=	1
w - 0.5p + v	=	0
w, p, u, v	$\geq$	0

$$
\begin{aligned}
&\widetilde{n} = 4 \\
&\widetilde{x} = (w, p, u, v)^T \\
&\widetilde{c} = (1, 0, 0, 0)^T \\
&\widetilde{b} = b \\
&\widetilde{A} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & -\frac{1}{2} & 0 & 1 \end{pmatrix}
\end{aligned}
$$

1.7. Struktur der zulässigen Menge

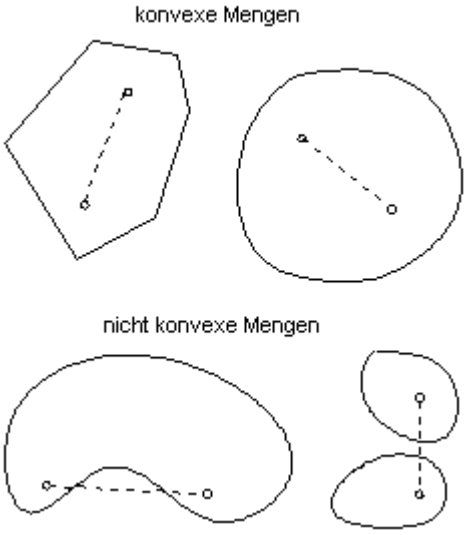
Wir gehen im folgenden von der erweiterten Standardform aus.

ZF:	$\mathbf{c}^T\mathbf{x}$	$\rightarrow$	max!
NB:	$\mathbf{A}\,\mathbf{x}$	=	$\mathbf{b}$
Nichtnegativität	$\mathbf{x}$	$\geq$	$\mathbf{0}$

Für den zulässigen Lösungsbereich gilt damit:

$$S = \{x \mid Ax = b, x \geq 0\} \qquad \text{Teilmenge aus } \mathbf{R}^n$$

Definition: Eine Teilmenge **C** des $\mathbf{R}^n$ heißt **konvex**, wenn mit zwei Punkten **x** und **y** auch die Verbindungsstrecke
$\lambda\,\mathbf{x}+(1-\lambda)\mathbf{y}$, $(0 \leq \lambda \leq 1)$ der Punkte zur Menge **C** gehört.



Theorem 1.1.:

Der Durchschnitt zweier konvexer Mengen ist ebenfalls konvex.

Daraus folgt: die zulässige Menge **S** des LP Programms in erweiterter Standardform ist ebenfalls konvex.

Beweis:

$S = \{x \mid Ax = b\} \cap \{x \mid x \geq 0\}$

$\{x \mid Ax = b\}$ ist konvex:

$x, y \in \{x|Ax = b\}$, also $Ax = b$ und $Ay = b$

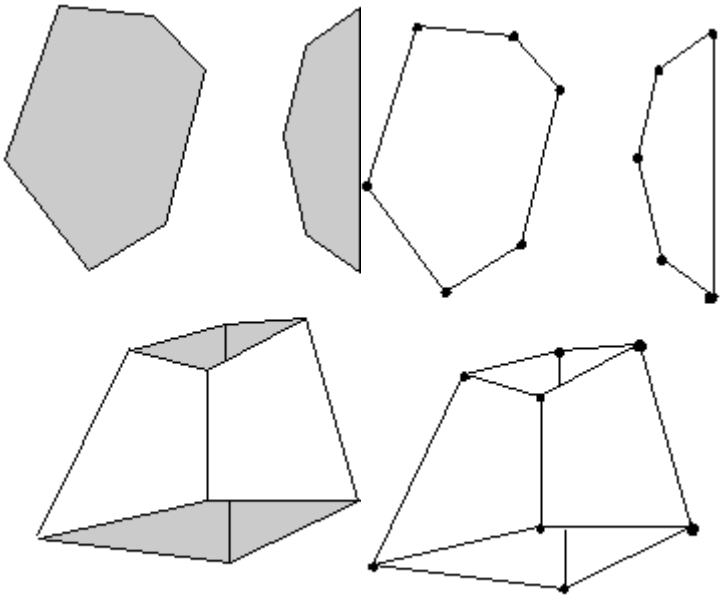
$\Longrightarrow A\ (\lambda\ x + (1 - \lambda)\ y) = \lambda\ A\ x + (1 - \lambda)\ A\ y = \lambda\ b + (1$

Also $\lambda x + (1 - \lambda)y \in \{x|Ax = b\}$

Jede Konvexkombination liegt also wieder in der Menge {x

$\{ x \mid x \geq 0\}$ ebenfalls konvex.
Wendet man den Satz von oben an, so zeigt sich, dass auch S k

Wegen der Linearität der Nebenbedingungen wird die zulässige Menge **S** eines LP stets von Hyperebenen begrenzt, ist also ein konvexer **Polyeder**.



• Extremalpunkte

Definition: Sei **C** eine konvexe Menge und **x** ∈ **C**. **x** heißt **Extremalpunkt** von C, wenn **x** nicht als Konvexkombination
λ **y**+(1-λ)**z,** (0 < λ < 1) von zwei anderen Punkten **y**, **z** ∈ **C** dargestellt werden kann.

Die Extremalpunkte eines (konvexen) Polyeders nennt man **Ecken**.

Lineare Programmierung aus "Linearer Algebra"-Sicht:

A ist eine m x n Matrix. Man kann stets annehmen, daß rg(**A**) = m (Rang von **A**), d.h. alle Zeilen sind linear unabhängig (m < n).
Denn wäre rg(**A**) < m, dann sind gewisse NB-Gleichungen redundant (oder sogar widersprüchlich). Man kann redundante NB solange weglassen, bis rg(**A**) = m erreicht ist.
Falls m = n, hätte wegen rg(**A**) = m das System **Ax = b** eine eindeutige Lösung, d.h. **S** besteht aus einem Punkt.

Neue Schreibweise: **A** = ($a_{.1}$, ... , $a_{.n}$), wobei $a_{.j}$ ($1 \leq j \leq n$) die Spaltenvektoren von **A** sind.
Da der rg(**A**) = m, gibt es unter $a_{.1}$, ... , $a_{.n}$ m Vektoren, die linear unabhängig sind.
Sei J die Menge aller m-elementigen Teilmengen $\{j_1, ..., j_m\} \subseteq \{1, ... , n\}$, sodaß $a_{.j1}$, ... , $a_{.jm}$ linear unabhängig sind.

Beispiel 1.7.

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & -1 \end{bmatrix} \qquad \text{rg}\,(A) = 2 \qquad b = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

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

Das Gleichungssystem **Ax = b** kann man auch anschreiben als:

$a_{.1}x_1 + \ldots + a_{.n}x_n = \mathbf{b}$ (*)

Sei $\{j_1, ..., j_m\} \in J$. Wir konstruieren eine spezielle Lösung von (*):
Sei $x_j := 0$ für $j \notin \{j_1, ..., j_m\}$. (*) reduziert sich dann zu

$a_{.j1}x_{j1} + \ldots + a_{.jm}x_{jm} = \mathbf{b}$ (**)

Da $a_{.j1}, \ldots, a_{.jm}$ laut Voraussetzung unabhängig sind, ist (**) eindeutig lösbar. Die Lösung sei $x_{j1}, \ldots, x_{jm}$
Damit hat man alle Variablen x_j bestimmt. Man nennt den entstehenden Vektor $\mathbf{x} = (x_1, \ldots, x_n)$ eine Basislösung von (*).

Beispiel 1.8. *Fortsetzung Beispiel 1.7.*

$\{1,3\} \in J$

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} x_1 + \begin{bmatrix} 1 \\ -1 \end{bmatrix} x_3 = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

$$\begin{aligned} x_1 + \quad & x_3 = 5 \\ 2x_1 - \quad & x_3 = 0 \end{aligned}$$

$$x_1 = \tfrac{5}{3}; \qquad x_3 = \tfrac{10}{3}; \qquad x_2 = 0$$

$$(\tfrac{5}{3}, 0, \tfrac{10}{3})^T \qquad \text{Basislösung des Gleichungssystems}$$

Es muß nicht unbedingt $\mathbf{x} \geq \mathbf{0}$ gelten, dh. eine Basislösung muß nicht unbedingt zulässig sein.
Gilt aber $\mathbf{x} \geq \mathbf{0}$, so spricht man von einer zulässigen Basislösung.
In diesem Fall heißen
die Variablen x_j mit $j \in \{j_1, ..., j_m\}$ Basisvariablen
die Variablen x_j mit $j \notin \{j_1, ..., j_m\}$ Nichtbasisvariablen

Beispiel 1.9.

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 4 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

{2,3,4} ∈ J
x_2, x_3, x_4 ... Basisvariable
x_1, x_5 ..... Nichtbasisvariable (d.h. $x_1 = 0$, $x_5 = 0$)

$$\begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} x_2 + \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} x_3 + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} x_4 = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

$$x_2 = 7, x_3 = 4, \boxed{x_4 = -2}$$

Basislösung, aber nicht zulässig (da $x_4 < 0$).

Andere Indexkombination: {1,2,4} ∈ J
x_1, x_2, x_4 ... Basisvariable
x_3, x_5 ..... Nichtbasisvariable (d.h. $x_3 = 0$, $x_5 = 0$)

$$\begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix} x_1 + \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} x_2 + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} x_4 = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

$$x_1 = 4, x_2 = 3, x_4 = 6, x_3 = 0, x_5 = 0$$

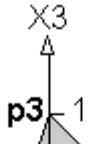
Das ist eine zulässige Basislösung!

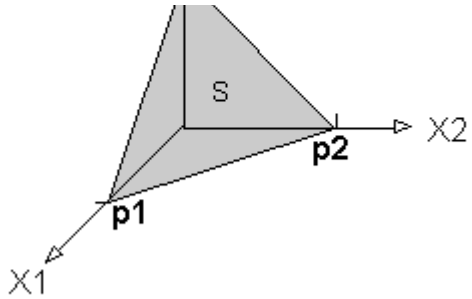
Theorem 1.2.:

Falls **S** nicht leer ist, gibt es mindestens eine zulässige Basislösung.

Verbindung zwischen geometrischen Begriffen und linearer Algebra:

Beispiel 1.10.
A = (1, 1, 1), **b** = 1, d.h. **S** = {**x** = (x_1, x_2, x_3) | $x_1 + x_2 + x_3 = 1$, x_1, x_2, $x_3 \geq 0$}.





Rg(A) = 1. J = { {1}, {2}, {3} }. Wir ermitteln die zulässigen Basislösungen:

{1}: $x_2 = 0,\ x_3 = 0,\ x_1 = 1$
{2}: $x_1 = 0,\ x_3 = 0,\ x_2 = 1$
{3}: $x_1 = 0,\ x_2 = 0,\ x_3 = 1$

Alle drei Basislösungen $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ sind zulässig.

Es sind gerade die Ecken des Polyeders **S**.

Theorem 1.3.: *Hauptsatz der linearen Optimierung*

Ein Punkt $\mathbf{x} \in \mathbf{S}$ ist genau dann Ecke des Polyeders **S**, falls **x** zulässige Basislösung von $\mathbf{Ax = b}$ ist.

Konsequenzen (falls **S** nicht leer):

- **S** hat mindestens eine Ecke.
- **S** hat höchstens $\begin{pmatrix} n \\ m \end{pmatrix}$ Ecken.

Definition: Die konvexe Hülle von k Punkten $x_1, \ldots, x_k$ ist die Menge aller Konvexkombinationen
$\lambda_1 x_1 + \ldots + \lambda_k x_k$ ($\lambda_i \geq 0, \lambda_1 + \ldots + \lambda_k = 1$), die sich aus $x_1, \ldots, x_k$ bilden lassen.

Beispiel 1.11. *Konvexe Hülle*

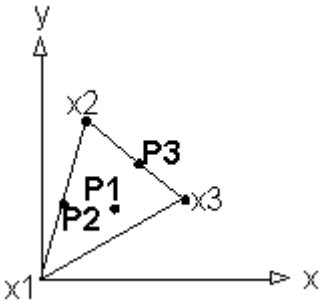
Konvexe Hülle von $\left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 4 \end{pmatrix}, \begin{pmatrix} 4 \\ 2 \end{pmatrix} \right\}$

enthält z.B. die Punkte P_1, P_2, P_3 :

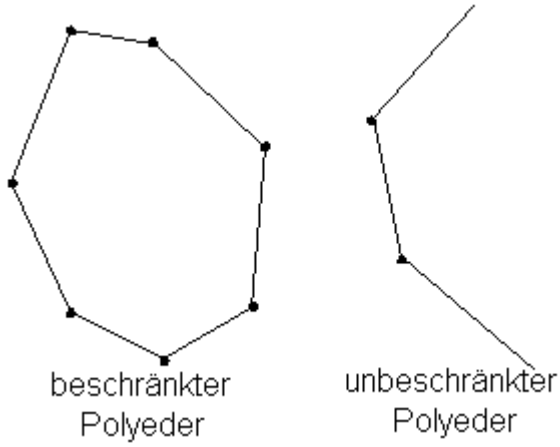
$$P_1 : \frac{1}{3}\begin{pmatrix} 0 \\ 0 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 1 \\ 4 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 4 \\ 2 \end{pmatrix} = \begin{pmatrix} \frac{5}{3} \\ 2 \end{pmatrix}$$

$$P_2 : \frac{1}{2}\begin{pmatrix} 0 \\ 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 \\ 4 \end{pmatrix} + 0\begin{pmatrix} 4 \\ 2 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 2 \end{pmatrix}$$

$$P_3 : 0\begin{pmatrix} 0 \\ 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 \\ 4 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 4 \\ 2 \end{pmatrix} = \begin{pmatrix} \frac{5}{2} \\ 3 \end{pmatrix}$$



Ein beschränkter Polyeder **S** ist stets die konvexe Hülle seiner Ecken.



Theorem 1.4.:
Für beschränktes **S** ist **S** die konvexe Hülle der zulässigen Basislösungen.

Theorem 1.5.: *(Lineare Programme werden an den Ecken gelöst)*
Das Optimum eines LP wird an einer Ecke (= zulässige Basislösung) angenommen.

Das Optimum eines LP wird an einer Ecke (= zulässige Basislösung) angenommen.
Genauer: ein Punkt $\mathbf{x} \in \mathbf{S}$ (**S** beschränkt) ist genau dann Lösung, wenn er Konvexkombination von Ecken ist, die ebenfalls Lösungen sind.

Beweis:
Angenommen der vorige Satz stimmt nicht, d.h. das Optimum wird in $\mathbf{x} \in \mathbf{S}$ angenommen und **x** ist weder Ecke noch Konvexkombination von optimalen Ecken. Wegen Theorem 1.4. läßt sich **x** als

$$\sum \lambda_i y^{(i)} \qquad (\lambda_i > 0, \sum \lambda_i = 1, \mathrm{y}^{(i)}\ \mathrm{Ecken})$$

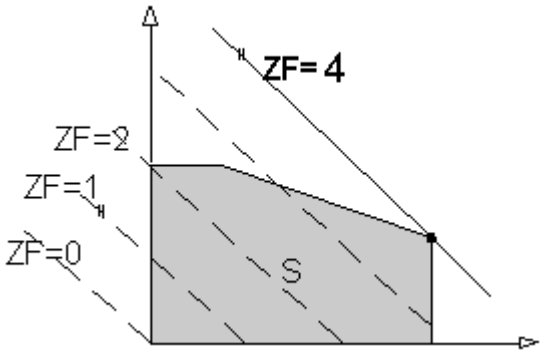
darstellen. Laut Voraussetzung sind nicht alle $y^{(i)}$ optimal, also $c^T y^{(i)} \leq c^T \mathbf{x}$, wobei für gewisse i sogar $c^T y^{(i)} < c^T \mathbf{x}$ gilt. Es folgt also

$$c^T x = c^T \sum \lambda_i y^{(i)} = \sum \lambda_i c^T y^{(i)} < \sum \lambda_i (c^T x) = (c^T x$$

und damit $c^T \mathbf{x} < c^T \mathbf{x}$ ein Widerspruch !
Somit kann die obige Annahme nicht gestimmt haben und die Aussage des Satzes gilt daher.

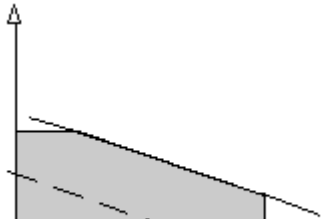
Anmerkung 1:

Die Lösung kann eindeutig sein oder auch nicht.
Beispiel:
Sei **S** = konvexe Hülle von $\{(0,0)^T, (3,0)^T, (3,1)^T, (1,2)^T, (0,2)^T\}$ und $\mathbf{c} = (1, 1)^T$, also ist die Zielfunktion $x_1 + x_2$
Man ermittelt die Niveaulinien, d.h. Linien deren Punkte gleichen ZF-Werte haben.
Z.B. Zielfunktionswert 0, d.h. $x_1 + x_2 = 0$ entspricht der Geraden $x_2 = -x_1$
oder Zielfunktionswert 1, d.h. $x_1 + x_2 = 1$ entspricht der Geraden $x_2 = -x_1 + 1$



Maximale Zielfunktion werden wir hier bekommen, wenn wir die Niveaulinie so weit wie möglich nach rechts oben verschieben. Die Lösung ist hier eindeutig gegeben und wird tatsächlich an einer Ecke von **S** angenommen.

Wählt man dagegen $\mathbf{c} = (1, 2)^T$, also Zielfunktion $x_1 + 2x_2$



ist die Lösung nicht mehr eindeutig. Die bestmögliche Niveaulinie schneidet den Bereich **S** auf der ganzen Strecke zwischen
$(3,1)^T$ und $(1,2)^T$. Alle diese Punkte liefern optimale Lösungen. Sie sind Konvexkombinationen der (ebenfalls optimalen) Ecken $(3,1)^T$ und $(1,2)^T$.

Anmerkung 2:
Obiger Satz liefert schon ein einfaches Lösungsverfahren zur Bestimmung der optimalen Lösung eines LP

- berechne **alle** zulässigen Basislösungen des LP
- stelle fest, welche davon den besten Zielfunktionswert haben

Problem: Das ist zu aufwendig. Die Anzahl der Basislösungen kann bis $\binom{n}{m}$ zu betragen.

Beispiel: Betrachtet man ein Problem mittlerer Größe mit n = 50 und m = 20, dann ist

$\binom{50}{20} = 4{,}71... \cdot 10^{13}$

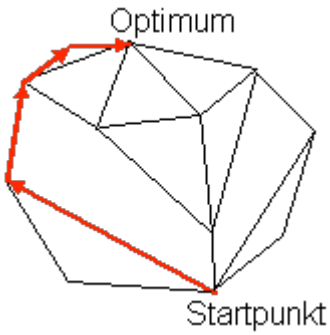
Geht man etwa von 1 Sekunde Rechenzeit pro Lösung aus, benötigt man rund 1,5 Millionen Jahre zum Finden des Optimums.
Selbst bei einer Rechenzeit von 0.00001 Sekunden pro Basislösung benötigt man noch 1500 Jahre.

Es ist daher nicht sinnvoll alle Ecken zu durchsuchen, sondern wenn man schon eine Ecke untersucht hat, nur mehr zu solchen mit besseren (oder zumindest mit nicht schlechteren) ZF-Werten weiter zu gehen.
Das ist die Grundidee des Simplex-Algorithmus.

2. Der Simplex Algorithmus

2.1. Grundlagen

Der Simplex-Algorithmus sucht das Optimum dadurch, daß es ausgehend von einer ersten zulässigen Lösung zu besseren oder zumindest gleich guten Lösungen fortschreitet.



Beispiel 2.1.: *Eine Firma produziert Glastüren und Holzfenster*

Benötigte Teilschritte pro Stück (Kapazität = Teilschritte pro Zeiteinheit):

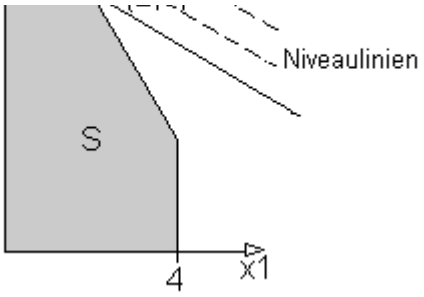
Maschine	Türen	Fenster	Kapazität
1	1	0	4
2	0	2	12
3	3	2	18
Gewinn	3	5	

- Gewinn, wenn nur Türen produziert werden: max. 4 Türen (pro Zeiteinheit), also Gewinn = 12
- Gewinn, wenn nur Fenster produziert werden: max. 6 Fenster (pro Zeiteinheit), also Gewinn = 30

Gesucht ist die gewinnmaximierende Produktionsmenge für Türen (x_1) und Fenster (x_2) .
Das ergibt als LP (formuliert in Standardform):

$$
\begin{aligned}
3\,x_1 + 5\,x_2 &\quad \rightarrow \quad \text{max!} \\
x_1 &\quad \leq \quad 4 \\
2\,x_2 &\quad \leq \quad 12 \\
3\,x_1 + 2\,x_2 &\quad \leq \quad 18 \\
x_1 , x_2 &\geq 0
\end{aligned}
$$

Graphische Darstellung der Standardform:



Niveaulinien: $3\,x_1 + 5\,x_2 = \text{const.}$

Durch Einführung von Schlupfvariablen (x_3 , x_4 , x_5) erhalten wir erweiterte Standardform:

$$
\begin{array}{lllll}
3\,x_1 + 5\,x_2 & & & \rightarrow & \max! \\
x_1 + x_3 & & = & 4 \\
x_2 + x_4 & & = & 12 \\
3x_1 + 2x_2 & + x_5 & = & 18 \\
x_1, \ldots, x_5 \geq 0 & & &
\end{array}
$$

n = 5, m = 3

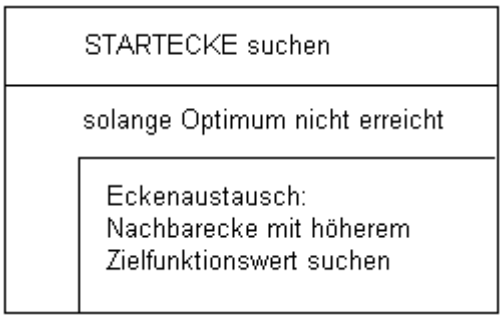
Die erweiterte Standardform kann graphisch nicht mehr dargestellt werden ($\mathbf{S} \subseteq \mathbf{R}^5$)

Man kann aber die zulässigen Basislösungen (Ecken) rechnerisch ermitteln (nicht ganz einfach, ziemliche Rechnerei):

(0,0,4,12,18)	ZF = 0	
(4,0,0,12,6)	ZF = 12	
(4,3,0,6,0)	ZF = 27	
(2,6,2,0,0)	ZF = 36	Optimum!
(0,6,4,0,6)	ZF = 30	

2.2. Lösen eines Linearen Programms

Grobschema des Simplex-Algorithmus

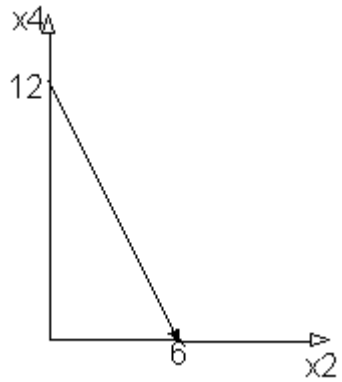


Zunächst benötigen wir irgendeine Ecke als Ausgangs- oder Startecke.
Im Beispiel 2.1. ist der Nullvektor Element von **S** und kann daher als Startecke herangezogen werden.
D.h. setze die Entscheidungsvariablen gleich 0 ($x_1 = 0$ und $x_2 = 0$), daraus folgen die Werte für die Schlupfvariablen unmittelbar ($x_3 = 4$, $x_4 = 12$ und $x_5 = 18$)

ZF:	[0]	$3x_1 + 5x_2$						$= 0$	
	[1]	x_1		$+ x_3$				$= 4$	
	[2]		$2x_2$		$+ x_4$			$= 12$	
	[3]	$3x_1 + 2x_2$				$+ x_5$		$= 18$	
		$x_1, ..., x_5 \geq 0$							
		Nichbasisvar.		Basisvar.					

Startecke ist daher: (0,0,4,12,18), Zielfunktionswert = 0.

Man kann den ZF-Wert sicherlich erhöhen, wenn man z.B. den Wert von x_2 erhöht (man könnte auch x_1 erhöhen, hätte aber "schwächeren Effekt"). Wegen der NB $2x_2 + x_4 = 12$ muß man als Kompensation für die Erhöhung von x_2 den Wert von x_4 entsprechend senken. Das kann man nur solange tun, so lange x_4 noch ≥ 0 ist.



das entspricht der Bewegung längs einer Kante von **S**!

Ist $x_4 = 0$ geworden, so hat man die nächste Ecke erreicht. Jetzt ist

Ist $x_4 = 0$ geworden, so hat man die nächste Ecke erreicht. Jetzt ist

- x_4 Nichtbasisvariable (= 0) und
- x_2 Basisvariable

d.h. man hat in der Basis x_4 durch x_2 ausgetauscht.
Rechnerisch geht das folgendermaßen:

- Koeffizient von x_2 in Zeile [2] auf 1 bringen
- x_2 durch x_4 ausdrücken, überall einsetzen
- Spalten für x_2 und x_4 vertauschen.

Also (Z aktueller Zielfunktionswert):

"Pivotzeile"	[2]	$2\,x_2 + x_4 = 12$
		$x_2 + 1/2\ x_4 = 6$
		$x_2 = -1/2\ x_4 + 6$
ZF-Zeile	[0]	$Z - 3\,x_1 - 5\,(-0.5\,x_4 + 6) = 0$
		$Z - 3\,x_1 + 5/2\,x_4 = 30$
Restliche Zeilen	[1]	$x_1 + x_3 = 4$
	[3]	$3\,x_1 + 2\,(-0.5\,x_4 + 6) + x_5 = 18$
		$3\,x_1 - x_4 + x_5 = 6$

Jetzt noch neu anschreiben, Spalten für x_2 und x_4 vertauschen:

[0]	Z	$-3\,x_1 + 5/2\,x_4$				$= 30$
[1]		x_1		$+ x_3$		$= 4$
[2]			$1/2\,x_4$		$+ x_2$	$= 6$
[3]		$3\,x_1 -$	x_4		$+ x_5$	$= 6$

Basis ist nun x_3 , x_2 , x_5
Mit den obigen Schritten haben wir erreicht, daß die Gleichungen wieder in analoger Form zu den Ausgangsgleichungen sind und daß der "rechte Teil" der linken Seite wieder die Einheitsmatrix als Koeffizientenmatrix hat. Setzt man die Nichtbasisvariablen x_1 und x_4 gleich Null, so kann man die neuene Basisvariablen x_3 , x_2 , x_5 und den Wert von Z sofort bestimmen: $Z = 30$, $x_3 = 4$, $x_2 = 6$, $x_5 = 6$
Das ist die Ecke (0,6,4,0,6).
Der Zielfunktionswert ist um 30 gestiegen!
Frage: Kann man ihn noch weiter erhöhen ?
Antwort: Vermutlich ja, da $Z = 3\,x_1 - 5/2\,x_4 + 30$. Die Zielfunktion wächst also, wenn x_1

zunimmt.
Keine Erhöhung des ZF-Wertes ist mehr möglich, wenn alle Koeffizienten von Nichtbasisvariablen in der ZF-Zeile ≥ 0 (auf die rechte Seite gebracht ≤ 0) sind. Die Variablen dürfen nämlich keine negativen Werte annehmen, also liefert Nullsetzen schon das Maximum !! Daher ist das **Optimum** erreicht, wenn in der Zielfunktions-Zeile [0] keine negativen Koeffizienten auftreten.

Nachdem wir x_2 als neue Basisvariable gewählt hatten, hatten wir die Zeile [2] $2\,x_2 + x_4 = 12$ als "**Pivotzeile**" gewählt.
Hätten wir die Zeile [3] $3\,x_1 + 2\,x_2 + x_5 = 18$ genommen, was wäre dann passiert ?

"Pivotzeile"	[3]	$3\,x_1 + 2\,x_2 + x_5 = 18$
		$3/2\,x_1 + x_2 + 1/2\,x_5 = 9$
		$x_2 = -3/2\,x_1 - 1/2\,x_5 + 9$
ZF-Zeile	[0]	$Z - 3\,x_1 - 5\,(-3/2\,x_1 - 1/2\,x_5 + 9) = 0$
		$Z + 9/2\,x_1 + 5/2\,x_5 = 45$
Restliche Zeilen	[1]	$x_1 + x_3 = 4$
	[2]	$2\,(-3/2\,x_1 - 1/2\,x_5 + 9) + x_4 = 12$
		$-3\,x_1 - x_5 + x_4 = -6$

In der letzten Gleichung wird die rechte Seite negativ !!
Das können wir aber nie erreichen, da wir ja $x_1 = x_5 = 0$ setzen wollen (neue Nichtbasisvariablen) und $x_4 \geq 0$ sein muß.
Man muß also die Wahl der Pivotzeile so vornehmen, daß die rechte Seite nichtnegativ bleibt. Das erreicht man durch folgende

Auswahlregel:

Bilde für jede Zeile, wo Koeffizienten von $x_j > 0$ sind, den Quotienten

$$\frac{\text{rechte Seite}}{\text{Koeffizient von } x_j}$$ (x_j auszutauschende Variable)

Wähle die Zeile so, daß dieser **Quotient minimal** wird!

Beispiel 2.1. Fortsetzung: Für obiges Beispiel würde das bedeuten:
In der ZF-Zeile nur mehr ein Koeffizient < 0, nämlich der von x_1. Also wird x_1 in die Basis genommen.
Berechne die Quotienten:
[1]: 4/1 = 4
[2]: Koeffizient von x_1 ist 0, scheidet daher aus
[3]: 6/3 = 2 <= das ist das Minimum, nimmt daher Zeile [3] als Pivotzeile
Es wird also x_5 aus der Basis herausgenommen.

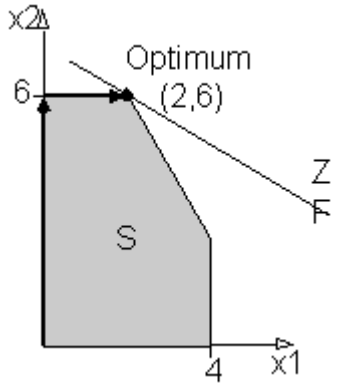
"Pivotzeile" [3] $x_2 = -1/2\,x_4 + 1/2\,x_5 = 2$

"Pivotzeile"	[3]	$x_1 - 1/3\ x_4 + 1/3\ x_5 = 2$
		$x_1 = 1/3\ x_4 - 1/3\ x_5 + 2$
ZF-Zeile	[0]	$Z - x_4 + x_5 - 6 + 5/2\ x_4 = 30$
		$Z + 3/2\ x_4 + x_5 = 36$
Restliche Zeilen	[1]	$1/3\ x_4 - 1/3\ x_5 + 2 + x_3 = 4$
		$1/3\ x_4 - 1/3\ x_5 + x_3 = 2$
	[2]	$1/2\ x_4 + x_2 = 6$

Jetzt noch neu anschreiben, Spalten für x_1 und x_5 vertauschen:

[0]	Z	$+ x_5 + 3/2\ x_4$			$= 36$
[1]		$-1/3\ x_5 + 1/3\ x_4$	$+ x_3$		$= 2$
[2]		$1/2\ x_4$	$+ x_2$		$= 6$
[3]		$1/3\ x_5 - 1/3\ x_4$		$+ x_1$	$= 2$

Die rechten Seiten der Nebenbedingungen sind wieder ≥ 0
Basis ist nun x_3 , x_2 , x_1
In der Zielfunktions-Gleichung haben jetzt alle x_j nichtnegative Koeffizienten. Wir sind also fertig!
Jetzt ist $x_5 = 0$ und $x_4 = 0$
und $x_3 = 2$, $x_2 = 6$, $x_1 = 2$,
Optimale Lösung ist also **x** = (2, 6, 2, 0, 0) und der **optimale Zielfunktionswert** ist **36**.
D.h. in der ursprünglichen Problemstellung: $x_1 = 2$ und $x_2 = 6$. Die Schlupfvariablen $x_3 = 2$, $x_4 = 0$, $x_5 = 0$.



2.3. Tabellarische Lösungsmethode

"Simplex Tableau" für das Beispiel 2.1.

ZF/Basisvariable	Koeffizienten von x_j					rechte Seite
	x_1	x_2	x_3	x_4	x_5	
Z	-3	-5	0	0	0	0
x_3	1	0	1	0	0	4
x_4	0	2	0	1	0	12
x_5	3	2	0	0	1	18

Tabelle für das allgemeine Problem:

ZF/BV	x_1	....	x_n	r.S.
Z	$a_{0\,1}$	....	$a_{0\,n}$	b_0
x_{r_1}	$a_{1\,1}$	....	$a_{1\,n}$	b_1
:	:		:	:
:	:		:	:
x_{r_m}	$a_{m\,1}$	....	$a_{m\,n}$	b_m

x_{r1}, ... , x_{rm} sind die Basisvariablen (zu Beginn: Schlupfvariable)

Ablauf des Algorithmus:

(1) Auswahl der **Pivotspalte j**: kleinstes $a_{0j} < 0$ suchen. Falls nur noch $a_{0j} \geq 0$, dann sind wir fertig.

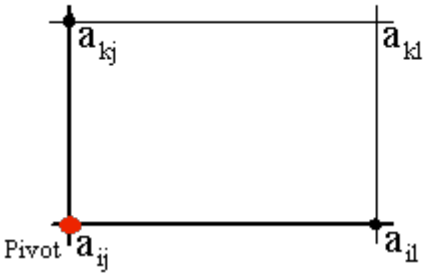
(2) Auswahl der **Pivotzeile i**: Jenes $i \geq 1$, für das b_i / a_{ij} minimal unter allen b_k / a_{kj} mit $a_{kj} > 0$.

(3) Ersetzen der Bezeichung x_{ri} (Var. der Pivotzeile) durch x_j (Var. der Pivotspalte) in der Spalte ZF/BV

(4) Verändern der Pivotzeile: $a_{i\,l}^{\text{neu}} = \frac{a_{i\,l}^{\text{alt}}}{a_{i\,j}^{\text{alt}}}$

Verändern der anderen Zeilen:

(5) $$a_{k\,l}^{\text{neu}} = \begin{cases} a_{k\,l}^{\text{alt}} - \frac{a_{k\,j}^{\text{alt}}}{a_{i\,j}^{\text{alt}}} a_{i\,l}^{\text{alt}} & \text{falls } l \neq j \\ 0 & \text{falls } l = j \end{cases}$$



Die Schritte (4) und (5) sind **elementare Umformungen** mit folgender Wirkung: Die neue Pivotspalte enthält 1 in der Pivotzeile und sonst lauter Nullen.

Beispiel 2.1. in Tableau-Schreibweise:

1. Schleifendurchlauf:

(1) Auswahl der **Pivotspalte j**: a_{02} = -5 ist minimal, also **j = 2**

(2) Auswahl der **Pivotzeile i**: 4/0 scheidet aus, 12/2 = 6 minimal, weil 18/2 = 9. Also **i = 2**

(3)

ZF/BV		x_1	$\mathbf{x_2}$	x_3	x_4	x_5	r.S.	
I	Z	-3	**-5**	0	0	0	0	I - (-5/2) III
II	x_3	1	**0**	1	0	0	4	II - (0/2) III
III	$\mathbf{x_4}$	**0**	**2**	**0**	**1**	**0**	**12**	III/2
IV	x_5	3	**2**	0	0	1	18	IV - (2/2) III

(4)+(5)

ZF/BV		x_1	$\mathbf{x_2}$	x_3	x_4	x_5	r.S.
I	Z	-3	**0**	0	5/2	0	30
II	x_3	1	**0**	1	0	0	4
III	$\mathbf{x_2}$	**0**	**1**	**0**	**1/2**	**0**	**6**
IV	x_5	3	**0**	0	-1	1	6

Von (0, 0) ausgehend haben wir die neue Ecke (0, 6) (x_1 = 0, x_2 = 6) mit Zielfunktionswert 30 erhalten.
Dies ist noch nicht das Optimum, da in der Zielfunktionszeile noch ein negativer Eintrag.

2. Schleifendurchlauf:

(1) Auswahl der **Pivotspalte j**: a_{01} = -3 ist minimal, also **j = 1**

(2) Auswahl der **Pivotzeile i**: 4/1 = 4, 6/0 scheidet aus, 6/3 = 2 ist minimal. Also **i = 3**

ZF/BV		$\mathbf{x_1}$	x_2	x_3	x_4	x_5	r.S.	
I	Z	**-3**	0	0	5/2	0	30	I - (-3/3) IV

(3)	II	x_3	**1**	0	1	0	0	4	II - (1/3) IV
	III	x_2	**0**	1	0	1/2	0	6	III - (0/3) IV
	IV	**x_5**	**3**	**0**	**0**	**-1**	**1**	**6**	IV/3

	ZF/BV		x_1	x_2	x_3	x_4	x_5	r.S.
	I	Z	**0**	0	0	3/2	1	36
(4)+(5)	II	x_3	**0**	0	1	1/3	-1/3	2
	III	x_2	**0**	1	0	1/2	0	6
	IV	**x_1**	**1**	**0**	**0**	**-1/3**	**1/3**	**2**

Von der Ecke (0, 6) ausgehend haben wir die Ecke (2, 6) ($x_1 = 2$, $x_2 = 6$) mit Zielfunktionswert 36 erhalten (mit Schlupfvariablen: (2, 6, 2, 0, 0))
Das ist schon das Optimum (kein negativer Eintrag mehr in der Zielfunktionszeile).

Frage: Wie sieht es aus, wenn das Optimum längs einer ganzen Kante auftritt ?

Beispiel 2.2.:

ZF $x_1 + x_2$ -> max!
NB $x_1 + x_2 \leq 1$
$x_1, x_2 \geq 0$

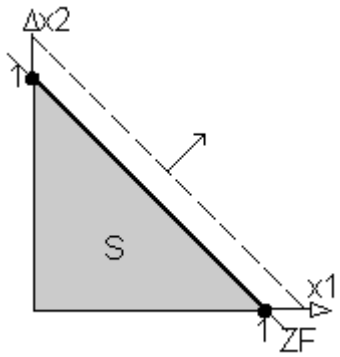


Tableau:

	x_1	x_2	x_3	r.S.
Z	-1	-1	0	0
x_3	1	1	1	1

Man kann hier x_1 oder x_2 austauschen und erhält $x_1 = 1$, $x_2 = x_3 = 0$, ZF = 1

	x_1	x_2	x_3	r.S.
Z	0	0	1	1
x	1	1	1	1

x_1	1	1	1	1

oder $x_2 = 1, x_1 = x_3 = 0,$ ZF = 1

	x_1	x_2	x_3	r.S.
Z	0	0	1	1
x_2	1	1	1	1

Der Simplex-Algorithmus liefert also jeweils nur **eine** der optimalen Ecken, nicht aber die "allgemeine" Lösung des Problems (in diesem Fall die Strecke zwischen (1, 0) und (0, 1)), die eine Konvexkombination aller optimalen Ecken ist.

Beispiel 2.3.: *Beispiel 1.3. mit Simplex Algorithmus*

w	$\rightarrow$	max!
w + p + u	=	1
w - 0.5p + v	=	0
w, p, u, v $\geq$ 0		

Lösung mit Hilfe des Simples Algorithmus

	w	p	u	v	r.S.
Z	-1	0	0	0	0
u	1	1	1	0	1
v	**1**	-1/2	0	1	0

Nur in 1.Spalte negativer Eintrag, daher j = 1.
1/1 = 1, 0/1 = 0 ist minimal, daher 2. Zeile

	w	p	u	v	r.S.
Z	0	-1/2	0	1	0
u	0	**3/2**	1	-1	1
w	1	-1/2	0	1	0

Der ZF Wert ist in diesem Schritt gleich geblieben.
2. Spalte negativ, daher j = 2.
Es gibt nur einen nicht-negativen Eintrag in diese Spalte, also wird diese Pivotzeile

	w	p	u	v	r.S.
Z	0	0	1/3	2/3	1/3
p	0	1	2/3	-2/3	2/3
w	1	0	1/3	4/3	1/3

Es ist also der optimale ZF Wert = 1/3, p = 2/3, w = 1/3 genauso wie im Kapitel 1.

Beispiel 2.4.:

Beispiel 2.4.:

$$1000\,x_1 + 200\,x_2 + 400\,x_3 \rightarrow \text{max!}$$
$$x_1 + x_2 + 8\,x_3 \leq 250$$
$$20x_1 + 2\,x_2 + x_3 \leq 200$$
$$x_1,\ x_2,\ x_3 \geq 0$$

In erweiterter Standardform:

$$1000\,x_1 + 200\,x_2 + 400\,x_3 \rightarrow \text{max!}$$
$$x_1 + x_2 + 8\,x_3 + x_4 = 250$$
$$20x_1 + 2\,x_2 + x_3 + x_5 = 200$$
$$x_1,\ x_2,\ x_3,\ x_4,\ x_5 \geq 0$$

Simplex-Tableau:

ZF/BV	x_1	x_2	x_3	x_4	x_5	r.S.
Z	-1000	-200	-400	0	0	0
x_4	1	1	8	1	0	250
x_5	**20**	2	1	0	1	200

Pivotspalte j = 1, da -1000 minimal.
Pivotzeile i = 2: 250/1 = 250, 200/20 = 10 ist minimal

ZF/BV	x_1	x_2	x_3	x_4	x_5	r.S.
Z	0	-100	-350	0	-50	10000
x_4	0	9/10	**159/20**	1	-1/20	240
x_1	1	1/10	1/20	0	1/20	10

Pivotspalte j = 3, da -350 minimal.
Pivotzeile i = 1: 240/(159/20) = 30.189 ist minimal, 10/(1/20) = 200

ZF/BV	x_1	x_2	x_3	x_4	x_5	r.S.
Z	0	-3200/53	0	7000/159	7600/159	1 090 000/53
x_3	0	6/53	1	20/159	-1/159	1600/53
x_1	1	**5/53**	0	-1/159	8/159	450/53

Pivotspalte j = 2, da -3200/53 der letzte negative Eintrag ist.
Pivotzeile i = 2: (1600/53)/(6/53) = 266.67, (450/53)/(5/53) = 90 ist minimal

ZF/BV	x_1	x_2	x_3	x_4	x_5	r.S.

ZF/BV	1	2	3	4	5	r.S.
Z	640	0	0	40	80	26000
x_3	-6/5	0	1	2/15	-1/15	20
x_2	53/5	1	0	-1/15	8/15	90

Also optimaler Zielfunktions-Wert: 26000
Lösung: $x_1 = x_4 = x_5 = 0$ (Nichtbasis-Variable) und $x_2 = 90$, $x_3 = 20$.

2.4. Dualität

Mit dem bisher entwickelten Verfahren können wir die "Normalaufgabe" des LP lösen

Standardform, wobei noch $\mathbf{b} \geq \mathbf{0}$:

$$\begin{array}{lll} \mathbf{c}^T\mathbf{x} & \rightarrow & \max! \\ \mathbf{A}\,\mathbf{x} & \leq & \mathbf{b} \\ \mathbf{x} & \geq & \mathbf{0} \end{array}$$

oder in erweiterter Standardform:

$$\begin{array}{lll} \mathbf{c}^T\mathbf{x} & \rightarrow & \max! \\ \mathbf{A}\,\mathbf{x} + \mathbf{I}\,\mathbf{u} & = & \mathbf{b} \\ \mathbf{x}, \mathbf{u} & \geq & \mathbf{0} \end{array}$$

(**I** Einheitsmatrix, **u** der Vektor der Schlupfvariablen)

Vorteil dieser "Normalaufgabe" ist, daß der **0**-Vektor eine zulässige Lösung ist, d.h. man spart sich die Suche nach einer Ausgangsecke.

Tableau's in Matrix-Schreibweise:

Starttableau:

Zielfunktions-Zeile: $\mathbf{t}^T = (\,-\mathbf{c}^T \mid \mathbf{0}^T \mid 0)$ eine Zeile
Nebenbedingungs-Zeilen: $\mathbf{T} = (\mathbf{A} \mid \mathbf{I} \mid \mathbf{b})$ m Zeilen

Im Beispiel 2.1. ist das Starttableau folgendermaß gegeben:

$\mathbf{t}^T = (\,-3\ \ -5 \mid 0\ 0\ 0 \mid 0)$

$$\mathbf{T} = \left(\begin{array}{cc|ccc|c} 1 & 0 & 1 & 0 & 0 & 4 \\ 0 & 2 & 0 & 1 & 0 & 12 \\ 3 & 2 & 0 & 0 & 1 & 18 \end{array}\right)$$

Endtableau:

Zielfunktions-Zeile: $\mathbf{t}^{*T} = (\,-\mathbf{c}^{*T} \mid \mathbf{y}^{*T} \mid \mathbf{Z}^*)$
Nebenbedingungs-Zeilen: $\mathbf{T}^* = (\mathbf{A}^* \mid \mathbf{S}^* \mid \mathbf{b}^*)$

Endtableau im Beispiel 2.1.:

$\mathbf{t}^{*T} = (\,0\ \ 0 \mid 0\ 3/2\ 1 \mid 36)$

$$\mathbf{T}^* = \left(\begin{array}{cc|ccc|c} 0 & 0 & 1 & 1/3 & -1/3 & 2 \\ 0 & 1 & 0 & 1/2 & 0 & 6 \\ 1 & 0 & 0 & -1/3 & 1/3 & 2 \end{array}\right)$$

Der Übergang von einem Tableau zum nächsten erfolgt durch Multiplikation mit einer

Transformationsmatrix.

Spalte j

$$\text{Zeile i} \quad \left(\begin{array}{ccccc|c|ccccc} 1 & 0 & .. & .. & 0 & -\frac{a_{1j}}{a_{ij}} & 0 & .. & .. & .. & 0 \\ 0 & 1 & .. & .. & 0 & -\frac{a_{2j}}{a_{ij}} & 0 & .. & .. & .. & 0 \\ .. & .. & .. & .. & .. & .. & .. & .. & .. & .. & .. \\ 0 & .. & .. & 0 & 1 & -\frac{a_{i-1,j}}{a_{ij}} & 0 & .. & .. & .. & 0 \\ \hline 0 & .. & .. & 0 & 0 & \frac{1}{a_{ij}} & 0 & .. & .. & .. & 0 \\ \hline 0 & .. & .. & 0 & 0 & -\frac{a_{i+1,j}}{a_{ij}} & 1 & 0 & .. & .. & 0 \\ .. & .. & .. & .. & .. & .. & .. & .. & .. & .. & .. \\ 0 & .. & .. & .. & 0 & -\frac{a_{n,j}}{a_{ij}} & 0 & .. & .. & 0 & 1 \end{array}\right) * \left(\begin{array}{ccc} a_{11} & .... & \\ .. & .... & \\ a_{m1} & .... & \end{array}\right.$$

Hier liefert das Produkt der k-ten Zeile mit der l-ten Spalte gerade die Transformationsregeln (4) und (5) des Simplex-Algorithmus (Die Zielfunktionszeile wurde hier ausgeklammert).
Sei **X** die Matrix, die sich als Produkt aller Transformationsmatrizen ergibt.
Dann läßt sich **T*** wie folgt darstellen: **T*** = **X T**

Die Matrix **X** läßt sich aus dem Endtableau bestimmen:
(**A*** | **S*** | **b***) = **X** (**A** | **I** | **b**)
Es gilt also **A*** = **X A,** **S*** = **X I,** **b*** = **X b**
und aus der 2. Gleichung ergibt sich **S*** = **X.**
Somit wird **T*** = (**A*** | **S*** | **b***) = (**S* A** | **S*** | **S* b**)

Ähnliches gilt für die ZF-Zeile: Sie war nie Pivotzeile. Mit ihr ist im Lauf des Verfahrens nichts anderes geschehen, als daß Vielfache anderer Zeilen von ihr abgezogen wurden, und diese Vielfachen sind wiederum Linearkombinationen der ursprünglichen Zeilen 1, ..., m.
Also: $\mathbf{t^{*T}} = \mathbf{t^T} - \mathbf{w^T T}$ oder $(-\mathbf{c^{*T}} \mid \mathbf{y^{*T}} \mid \mathbf{Z}^*) = (-\mathbf{c^T} \mid \mathbf{0^T} \mid 0) - \mathbf{w^T}(\mathbf{A} \mid \mathbf{I} \mid \mathbf{b})$
und daher $\mathbf{c^{*T}} = -\mathbf{c^T} - \mathbf{w^T A}$, $\mathbf{y^{*T}} = -\mathbf{w^T I}$, $\mathbf{Z}^* = -\mathbf{w^T b}$
Aus der 2. Gleichung ergibt sich **w = - y*** und somit wird
$\mathbf{t^{*T}} = (-\mathbf{c^T} + \mathbf{y^{*T}A} \mid \mathbf{y^{*T}} \mid \mathbf{y^{*T}\, b})$
Insgesamt hat also das Endtableau die Form:

$$\left(\begin{array}{c|c|c} -c^T + y^{*T}A & y^{*T} & y^{*T}b \\ S^*A & S^* & S^*b \end{array}\right)$$

Da dies das Tableau ist, für das die Abbruchbedingung (ZF-Zeile nicht-negativ) erfüllt ist, gilt:

1) $-\mathbf{c^T} + \mathbf{y^{*T}\, A} \geq \mathbf{0^T}$, also $\mathbf{y^{*T}\ A} \geq \mathbf{c^T}$ oder $\mathbf{A^T\ y^*} \geq \mathbf{c}$
und
2) $\mathbf{y^{*T}} \geq \mathbf{0^T}$, also $\mathbf{y^*} \geq \mathbf{0.}$

y* ist also sicher eine **zulässige** Lösung des folgenden LP:

$\mathbf{b^T y} \quad \rightarrow$ min!

$$\begin{array}{lll} \mathbf{A}^T\mathbf{y} & \geq & \mathbf{c} \\ \mathbf{y} & \geq & \mathbf{0} \end{array}$$

Dies nennt man das (zur Normalaufgabe) **duale LP**.

$$\left\{ \begin{array}{lll} c^T x & = & \max! \\ A x & \leq & b \\ x & \geq & 0 \end{array} \right. \qquad \left\{ \begin{array}{lll} b^T y & = & \min! \\ A^T y & \geq & c \\ y & \geq & 0 \end{array} \right.$$

(P): primales Programm (D): duales Programm

Die Koeffizienten der Zielfunktion bzw. der Nebenbedingungen haben ihre Rollen vertauscht! Aus max! wird min!, aus $\leq$ wird $\geq$, die Matrix **A** wird transponiert.

Achtung: Da **A** im allgemeinen keine quadratische Matrix ist, ändert sich auch die Anzahl der Zeilen und Spalten. Die Vektoren **x** und **y** haben verschiedene Anzahl von Komponenten !

Frage: Was ist das duale Programm zum dualen Programm ?

Formt (D) in Standardform um:

$$\begin{array}{lll} -\mathbf{b}^T\mathbf{y} & \rightarrow & \max! \\ -\mathbf{A}^T\mathbf{y} & \leq & -\mathbf{c} \\ \mathbf{y} & \geq & \mathbf{0} \end{array}$$

und dualisiert:

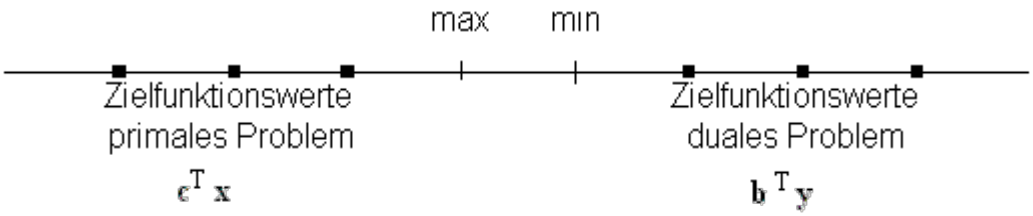
$$\begin{array}{lll} -\mathbf{c}^T\mathbf{z} & \rightarrow & \min! \\ -\mathbf{A}\mathbf{z} & \geq & -\mathbf{b} \\ \mathbf{z} & \geq & \mathbf{0} \end{array} \quad \text{oder} \quad \begin{array}{lll} \mathbf{c}^T\mathbf{z} & \rightarrow & \max! \\ \mathbf{A}\mathbf{z} & \leq & \mathbf{b} \\ \mathbf{z} & \geq & \mathbf{0} \end{array}$$

Das ist wieder genau das primale Problem (P)
D.h. das duale Problem zum dualen Problem ist wieder das primale Problem.

Theorem 2.1.:
Ist **x** zulässig für (P) und **y** zulässig für (D), so gilt $\mathbf{c}^T\mathbf{x} \leq \mathbf{b}^T\mathbf{y}$.

Beweis:
$\mathbf{A}\mathbf{x} \leq \mathbf{b}$ und $\mathbf{A}^T\mathbf{y} \geq \mathbf{c}$ bzw. $\mathbf{y}^T\mathbf{A} \geq \mathbf{c}^T$. Also gilt $\mathbf{c}^T\mathbf{x} \leq (\mathbf{y}^T\mathbf{A})\,\mathbf{x} = \mathbf{y}^T(\mathbf{A}\mathbf{x}) \leq \mathbf{y}^T\mathbf{b} = \mathbf{b}^T\mathbf{y}$
(da $\mathbf{y}^T\mathbf{b}$ eine (1x1) Matrix, bleibt es beim Transponieren gleich!)

Es muß daher das Maximum der zulässigen Werte von (P) (= die Lösung von (P)) kleiner gleich dem Minimum der zulässigen Werte von (D) (= die Lösung von (D)) sein.



Es stellt sich sogar heraus: Wenn (P) und (D) überhaupt optimale Lösungen haben, so gilt Gleichheit (d.h. es entsteht keine Lücke)

Theorem 2.2.:
Ist **x*** optimal für (P) und **y*** optimal für (D), so gilt $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}^*$.

Beweis:
Wir haben gesehen: Falls die optimale Lösung von (P) existiert, sieht die ZF-Zeile des letzten Tableaus so aus:
$(-\mathbf{c}^T + \mathbf{y}^{*T}\mathbf{A} \mid \mathbf{y}^{*T} \mid \mathbf{y}^{*T}\mathbf{b})$ (wobei hier noch nicht sicher ist, daß **y*** die Lösung von (D) ist)
und es gilt: $\mathbf{A}^T\mathbf{y}^* \geq \mathbf{c}$ und $\mathbf{y}^* \geq \mathbf{0}$, d.h. **y*** ist zulässige Lösung von (D).
Den optimalen ZF-Wert von (P) kann man aber aus der ZF-Zeile sofort ablesen,
er ist $\mathbf{y}^{*T}\mathbf{b} = \mathbf{b}^T\mathbf{y}^*$!!
Also gilt $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}^*$. Da aber $\mathbf{c}^T\mathbf{x}^* \leq \mathbf{b}^T\mathbf{y}$ für jedes zulässige **y**, so ist **y*** optimal für (D), also gilt auch:
$\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}^*$.

y* heißt **duale Lösung** zu **x***.

Wie man aus dem obigen Beweis sieht, kann man die duale Lösung aus dem Endtableau des auf (P) angewendeten Simplex-Algorithmus einfach ablesen!

Beispiel 2.5.: Ein Unternehmen benötigt 3 Mineralien M1, M2, M3 zur Weiterverarbeitung. Diese werden vom Unternehmen selbst in einem Separationsprozeß aus den beiden Rohstoffen R1 und R2 gewonnen.

Anteilswerte der Ausbeute an Mineralien aus den Rohstoffen:

	M1	M2	M3
R1	0.03	0.125	0.4
R2	0.6	025	0.05

Benötigte Mengen pro Monat in Tonnen:

M1	30
M2	25
M3	20

Entstehungspreise pro Tonne:

R1	250
R2	200

y_1 = pro Monat gekaufte Menge von R1 (in Tonnen)
y_2 = pro Monat gekaufte Menge von R2 (in Tonnen)

Zielfunktion:	$250\ y_1 + 200\ y_2$	$\rightarrow$	min!
Nebenbedingungen:	$0.03\ y_1 + 0.6\ y_2$	$\geq$	30
	$0.125\ y_1 + 0.25 y_2$	$\geq$	25
	$0.4\ y_1 + 0.05\ y_2$	$\geq$	20

Vorzeichenbedingungen: $y_1 \geq 0,\ y_2 \geq 0$
Durch Multiplikation ein bißchen verschönert:

Zielfunktion:	$250\ y_1 + 200\ y_2$	$\rightarrow$	min!
Nebenbedingungen:	$y_1 + 20\ y_2$	$\geq$	1000
	$y_1 + 2 y_2$	$\geq$	200
	$8\ y_1 + y_2$	$\geq$	400

Vorzeichenbedingungen: $y_1 \geq 0,\ y_2 \geq 0$

Man könnte dieses LP sofort in Standardform umformen und den Simplex-Algorithmus starten, hätte dabei aber das Problem, daß die rechten Seiten der NB negativ werden und daher der Nullvektor keine zulässige Lösung ist.
Es ist hier vorteilhaft zu "dualisieren" oder (was wie wir gesehen haben dasselbe ist) unser Problem als duales Programm aufzufassen und das zugehörige primale Programm zu suchen.

Dazu setzen wir: $\mathbf{b}^T = (250, 200)$, $\mathbf{A}^T = \begin{pmatrix} 1 & 20 \\ 1 & 2 \\ 8 & 1 \end{pmatrix}$, $\mathbf{c} = \begin{pmatrix} 1000 \\ 200 \\ 400 \end{pmatrix}$

Primales Programm dazu ist dann:

Zielfunktion:	$1000\ x_1 + 200\ x_2 + 400\ x_3$	$\rightarrow$	max!
Nebenbedingungen:	$x_1 + x_2 + 8\ x_3$	$\leq$	250
	$20\ x_1 + 2\ x_2 + x_3$	$\leq$	200

Vorzeichenbedingungen: $x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0$

Das ist eine Normalaufgabe, da $250 \geq 0$ und $200 \geq 0$. Die Lösung mit dem Simplex-Algorithmus wurde bereits im Beispiel 2.4. berechnet.
Als Zielfunktionszeile im letzten Tableau erhält man:

	Entscheidungsvariable			Schlupfvar.		
	x_1	x_2	x_3	x_4	x_5	r.S.
Z	640	0	0	40	80	26000

$-c^T + y^{*T} A \qquad y^{*T} \qquad y^{*T} b$

Daraus kann man die duale Lösung (d.h. die Lösung unseres Problems) und den ZF-Wert sofort ablesen:

$\mathbf{y}^{*T} = (40, 80)$ und $\mathbf{b}^T\mathbf{y}^* = 26000$.

Interpretation der dualen Variablen ("Sensitivität")

Eine Nebenbedingung heißt **aktiv**, falls sie mit Gleichheit erfüllt ist. Andernfalls heißt sie **passiv**.

Aus Beispiel 2.1.betrachten wir die (nicht optimale) Ecke (0,6,4,0,6):

Entscheidungsvariablen:

$x_1 \geq 0$	aktiv (da $x_1 = 0$)
$x_2 \geq 0$	passiv

Schlupfvariablen (entsprechen den Nebenbedingungen):

$x_3 \geq 0$	$(x_1 \leq 4)$	passiv
$x_4 \geq 0$	$(2x_2 \leq 12)$	aktiv (da $x_4 = 0$, d.h. auch $2x_2 = 12$)
$x_5 \geq 0$	$(3x_1 + 2x_2 \leq 18)$	passiv

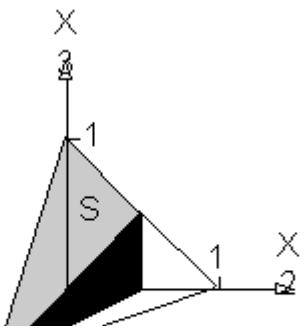
Sei allgemein $x_1, \ldots, x_n$ eine zulässige Basislösung zur Indexmenge $\{j_1, \ldots, j_m\}$.
Dann gilt für alle Nichtbasis-Variablen: $x_i = 0$, falls $i \notin \{j_1, \ldots, j_m\}$
und im allgemeinen für Basis-Variablen: $x_i \neq 0$, falls $i \in \{j_1, \ldots, j_m\}$ (das muß aber nicht gelten)

D.h. die zu Nichtbasis-Variablen gehörigen Bedingungen sind immer aktiv. Die zu Basis-Variablen gehörigen sind i.a. passiv, können in Sonderfällen aber auch aktiv sein.

Definition:
Eine Ecke heißt **degeneriert**, falls es mindestens eine Basisvariable mit einer aktiven Bedingung gibt (d.h. mindestens eine Basisvariable hat den Wert 0)

Beispiel 2.6.:
Zulässiger Bereich definiert durch folgende Ungleichungen:

$$x_1 + x_2 + x_3 \leq 1$$
$$x_1 + 2x_2 \leq 1$$
$$x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0$$



In erweiterter Standardform:

$x_1 + x_2 + x_3 + x_4 \leq 1$

$x_1 + 2\,x_2 + x_5 \leq 1$

$x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0,\ x_5 \geq 0$

Wähle als Indexmenge $\{j_1, j_2\} = \{1, 2\}$

$x_1 + x_2 = 1$

$x_1 + 2\,x_2 = 1$

Basislösung dazu: $x_1 = 1$, $x_2 = 0$ -> degeneriert!

In der Ecke (1,0,0,0,0) sind nicht nur die Bedingungen $x_3 \geq 0$, $x_4 \geq 0$, $x_5 \geq 0$ aktiv (das muß sein, da sie Nichbasis-Variablen sind), sondern auch die Bedingung $x_2 \geq 0$.

"Zufälligerweise" erfüllt diese Ecke, für die die beiden NB und $x_3 \geq 0$ mit Gleichheit erfüllt sind auch noch $x_2 \geq 0$ mit Gleichheit.

Solche Fälle sind aber eher selten. Im folgenden werden wir daher von ihnen absehen.

Theorem 2.3.:

Sei **x*** eine nicht-degenerierte Lösung und **y*** die duale Lösung. Dann gilt für Vektoren $\tilde{b}$, die hinreichend nahe beim Vektor **b** liegen, daß das "gestörte" Problem

$\mathbf{c}^T\mathbf{x} \rightarrow \max!$

$\mathbf{A}\,\mathbf{x} \leq \tilde{b}$

$\mathbf{x} \geq \mathbf{0}$

den Optimalwert $\tilde{b}^T\mathbf{y}^*$ besitzt. (Das ungestörte Problem besitzt Optimalwert $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}^*$)

Beweis:

Endtableau:

$$\left(\begin{array}{c|c|c} -c^T + y^{*T}A & y^{*T} & y^{*T}b \\ S^*A & S^* & S^*b \end{array}\right)$$

Da **x*** nicht degenereriert, haben alle Basisvariablen Werte > 0, d.h. die rechte Seite der NB ist > 0, also $\mathbf{S}^*\,\mathbf{b} > 0$.

Dann gilt aber auch $\mathbf{S}^*\,\tilde{b} > 0$ für $\tilde{b}$ hinreichend nahe bei **b**.Somit ist

$$\left(\begin{array}{c|c|c} -c^T + y^{*T}A & y^{*T} & y^{*T}\widehat{b} \\ S^*A & S^* & S^*\widehat{b} \end{array}\right)$$

wieder ein gültiges Simplex-Tableau, und zwar offenbar eines zu ZF und NB des gestörten

wieder ein gültiges Simplex-Tableau, und zwar offenbar eines zu ZF und NB des gestörten Problems (sieht man z.B. indem man den untern Teil mit $\mathbf{S^{*-1}}$ multipliziert).
Die ZF-Zeile erfüllt noch immer die Abbruchbedingung, also ist **y*** optimale Lösung des entsprechenden dualen Problems und $\mathbf{y^{*T}}\tilde{\mathbf{b}} = \tilde{\mathbf{b}}^{T}\mathbf{y^{*}}$ optimaler ZF-Wert sowohl für das primale wie auch für das duale Problem.

Folgerung:

Leitet man den optimalen ZF-Wert $\tilde{\mathbf{b}}^{T}\mathbf{y^{*}}$ nach $\tilde{\mathbf{b}}_i^{T}$ ab, so erhält man folgendes:

$$\frac{\partial}{\partial \tilde{b}_i} \tilde{b}^T y^* = \frac{\partial}{\partial \tilde{b}_i} \sum_{j=1}^{m} \tilde{b}_j y_j^* = \frac{\partial}{\partial \tilde{b}_i} (\tilde{b}_i y_i^*) = y_i^*$$

Also: Die i-te Komponente der dualen Lösung ist die Ableitung des Optimalwerts nach dem i-ten Koeffizienten auf der rechten Seite.
$\mathbf{y}_i^{*}$ gibt an, wie "sensitiv" die optimale Lösung auf Veränderung von b_i reagiert.

Im Beispiel 2.5:
Die duale Lösung lautet (40, 80).
Läßt man in der ersten NB eine Erhöhung von 250 auf 251 zu, so dann der maximale ZF-Wert um 40 gesteigert werden.
Läßt man in der zweiten NB eine Erhöhung von 200 auf 201 zu, so dann der maximale ZF-Wert um 80 gesteigert werden.
Analog für Senkungen der rechten Seiten.

In der ökonomischen Literatur nennt man diese Werte **Schattenpreise** ("Um wieviel niedriger ist mein Gewinn, wenn eine meiner Nebenbedingungen um einer Einheit strenger wird?" bzw. "Um wieviel höher ist mein Gewinn, wenn einer meiner Nebenbedingungen um eine Einheit lockerer wird?").

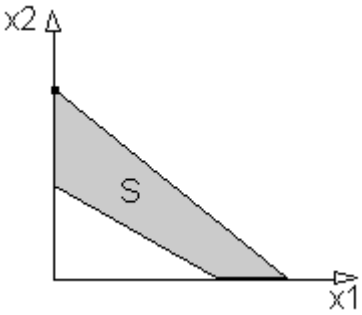
2.5. Ergänzungen zum Simplexalgorithmus

A. Negative rechte Seiten

Wir hatten bisher bei der Normalaufgabe $\mathbf{b} \geq 0$ vorausgesetzt. Was ist nun zu tun, wenn ein oder mehrere $b_i < 0$ sind ?

Beispiel 2.7.:

$$
\begin{array}{lcl}
x_1 + 3\,x_2 & \rightarrow & \max! \\
-x_1 - 2\,x_2 & \leq & -2 \\
x_1 + x_2 & \leq & 3 \\
x_1 \geq 0,\ x_2 \geq 0 & &
\end{array}
$$



Der Nullvektor ist in diesem Fall keine zulässige Lösung mehr, kann daher als Startecke nicht verwendet werden !

Lösungsansatz: **Groß-M-Methode**

Bringe das Problem zunächst in Erweiterte Standardform:

$$
\begin{array}{lcl}
x_1 + 3\,x_2 & \rightarrow & \max! \\
-x_1 - 2\,x_2 + u_1 & = & -2 \\
x_1 + x_2 + u_2 & = & 3 \\
x_1,\ x_2,\ u_1,\ u_2 \geq 0 & &
\end{array}
$$

Wenn man nun die erste NB wieder mit (-1) multipliziert, erhält man eine positive rechte Seite, aber der Koeffizient von u_1 ist gleich -1 und damit negativ. Für die Basisvariablen im Starttableau braucht man aber Koeffizienten (+1).

Trick: Man führt eine zusätzliche "künstliche Variable" v_1 ein (bei mehreren neg. rechten Seiten, mehrere Variable v_i).
Erhält nun als Gleichung: $x_1 + 2\,x_2 - u_1 + v_1 = 2$
v_1 hat nun als Koeffizienten (+1) in dieser Gleichung und Koeffizient 0 in allen anderen, ist daher als Basisvariable geeignet für das Starttableau.

v_1 sollte letztlich 0 werden. Wie erreicht man das ?

Werte $v_1 > 0$ sind in der ZF so stark zu "bestrafen", daß sie sich in der optimalen Lösung nicht ergeben werden. Das nennt man die **Groß-M-Methode**
Die ZF wird also geändert auf $x_1 + 3\,x_2 - Mv_1 = \max!$
Falls $v_1 = 0$, so hat man die ursprüngliche ZF, andernfalls ist die Strafzahlung $- Mv_1$ für die Verletzung von $v_1 = 0$ so stark, daß keine Chance besteht dies als optimale Lösung zu erhalten.
Man kann M allgemein lassen und als "sehr groß" voraussetzen.

$$
\begin{aligned}
x_1 + 3\,x_2 \qquad\qquad - Mv_1 \qquad\qquad &\rightarrow \max! \\
x_1 + 2\,x_2 - u_1 \qquad + v_1 \qquad\qquad &= 2 \\
x_1 + x_2 \qquad + u_2 \qquad\qquad &= 3 \\
x_1,\ x_2,\ u_1,\ u_2,\ v_1 &\geq 0
\end{aligned}
$$

Wir haben nun ein Tableau, das in der Spalte einer Basisvariablen (v_1) einen Wert $\neq 0$ hat. Damit aber v_1 Basisvariable werden kann, brauchen wir dort 0. Das können wir erreichen, indem wir das M-fache der ursprünglichen Nebenbedingung zur Zielfunktion addieren:

$$
\begin{aligned}
Z &= x_1 + 3\,x_2 \qquad - Mv_1 \\
2M &= Mx_1 + 2M\,x_2 - Mu_1 + Mv_1
\end{aligned}
$$

das ergibt: $Z + 2M = (M+1)x_1 + (2M + 3)\,x_2 - Mu_1$
bzw. $Z - (M+1)x_1 - (2M + 3)\,x_2 + Mu_1 = -2M$

ZF/BV		x_1	$\mathbf{x_2}$	u_1	u_2	v_1	r.S.	
I	Z	-M-1	**-2M-3**	M	0	0	-2M	I + (M+3/2) II
II	$\mathbf{v_1}$	**1**	**2**	**-1**	**0**	**1**	**2**	II/2
III	u_2	1	**1**	0	1	0	3	III - (1/2) II

ZF/BV		x_1	x_2	$\mathbf{u_1}$	u_2	v_1	r.S.	
I	Z	1/2	0	**-3/2**	0	M+3/2	3	I + 3 III
II	x_2	1/2	1	**-1/2**	0	1/2	1	II+III
III	$\mathbf{u_2}$	**1/2**	**0**	**1/2**	**1**	**-1/2**	**2**	2III

ZF/BV		x_1	x_2	u_1	u_2	v_1	r.S.
I	Z	2	0	0	3	M	9
II	x_2	1	1	0	1	0	3
III	u	1	0	1	2	-1	4

III u_1 1 0 1 2 -1 4

Wir erhalten also: $x_2 = 3$, $u_1 = 4$, $x_1 = 0$, $u_2 = 0$, $v_1 = 0$.

Lösung ist also: $(0, 3)^T$
Zielfunktionswert ist dann $x_1 + 3\,x_2 = 0 + 3.3 = 9$

Fortsetzung Beispiel 0.3.:

$m + 5v + 7b \rightarrow \max!$
$m + v + b = 1$
$m \geq 0.2$
$m + v - b \geq 0$
$-1/3m - 1/3v + b \geq 0$
$m, v, b \geq 0$

Mit Hilfe der ersten NB kann man b durch m und v ausdrücken: $b = 1 - m - v$
(Vorsicht! Es muß $b \geq 0$ bleiben !!)

$-6m - 2v \rightarrow \max!$
$-m \leq -0.2$
$-2m - 2v \leq -1$
$2/3m + 2/3v \leq 1$
$m + v \leq 1$ (da $b \geq 0$ sein muss. Diese NB ist aber redundant, siehe vorherige NB)
$m, v \geq 0$

Brauche also 2 künstliche Variablen z_1, z_2

$-6m - 2v - Mz_1 - Mz_2 \rightarrow \max!$
$m - u_1 + z_1 = 0.2$
$2m + 2v - u_2 + z_2 = 1$
$2/3m + 2/3v + u_3 = 1$
$m + v \leq 1$ (da $b \geq 0$ sein muss)
$m, v, u_1, u_2, u_3, z_1, z_2 \geq 0$

1. und 2. NB mit M multiplizieren und zu ZF addieren. Ergibt neue ZF:
$(3M-6)m + (2M-2)v - Mu_1 - Mu_2 = \max!$

Simplex-Tableau:

	m	v	u_1	u_2	u_3	z_1	z_2	r.S.
Z	-3M+6	-2M+2	M	M	0	0	0	1.2M
z_1	1	0	-1	0	0	1	0	0.2
z_2	2	2	0	-1	0	0	1	1
u_3	2/3	2/3	0	0	1	0	0	1

	m	v	u_1	u_2	u_3	z_1	z_2	r.S.
Z	0	-2M+2	-2M+6	M	0	3M-6	0	0.6M-1.2
m	1	0	-1	0	0	1	0	0.2
z_2	0	2	2	-1	0	-2	1	0.6
u_3	0	2/3	2/3	0	1	-2/3	0	13/15

	m	v	u_1	u_2	u_3	z_1	z_2	r.S.
Z	0	0	4	1	0	M-4	M-1	1.2M-1.8
m	1	0	-1	0	0	1	0	0.2
v	0	1	1	-1/2	0	-1	1/2	0.3
u_3	0	0	0	4/3	1	0	-1/3	2/3

Lösung ist also: m = 0.2, v = 0.3 (also b = 0.5)

B. Unlösbarkeit

In gewissen Fällen sind die Nebenbedingungen miteinander unverträglich, d.h. S = $\emptyset$.
Wie erkennt man das in einem Lösungsverfahren ?

Theorem 2.4.:
Das ursprünglichen Problem hat genau dann keine zulässige Lösung, wenn in der Lösung des modifizierten Problems nach der Groß-M-Methode mindestens eine künstliche Variable einen Wert > 0 hat.

Beispiel 2.7.:

x_1 -> max!
$x_1 \geq 5$
$x_1 \leq 3$ Hier widersprechen sich die Nebenbedingungen!

$x_1 - M\, v_1$ -> max!
$x_1 - u_1 + v_1 = 5$
$x_1 + u_2 = 3$

Daraus erhalten wir:
$u_1 = x_1 + v_1 - 5$
$u_2 = -x_1 + 3$

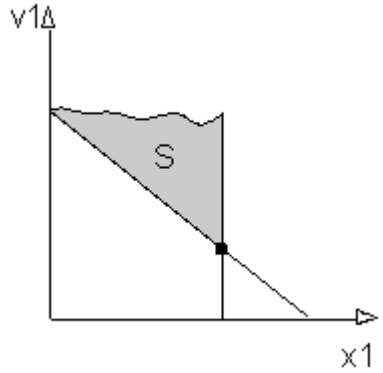
und somit:

$x_1 - M\, v_1$ -> max!

$x_1 + v_1 \rightarrow \max!$

$x_1 + v_1 \geq 5$ (wegen $u_1 \geq 0$)

$-x_1 \geq -3$ (wegen $u_2 \geq 0$)



Als Lösung erhalten wir $x_1 = 3$, $v_1 = 2$. Es ist also $v_1 > 0$ und daher ist das ursprüngliche Problem unlösbar.

C. Unbeschränktheit

Wenn der optimale ZF-Wert bei ∞ (für ein Maximierungsproblem) bzw. bei -∞ (für ein Minimierungsproblem) liegt.

Ein ganz einfaches Beispiel:

$x_1 \rightarrow \max!$

$x_1 \geq 0$

Wie erkennt man nun diese Situation im Simplex-Algorithmus ?

Theorem 2.5.:

Falls $a_{ij} \leq 0$ für alle (j der Index der Pivotspalte), so ist das Lineare Programm unbeschränkt (Auswahl in der Pivotzeile nicht möglich).

Beispiel 2.8.:

$2x_1 + x_2 \rightarrow \max!$

$-x_1 + x_2 \leq 2$

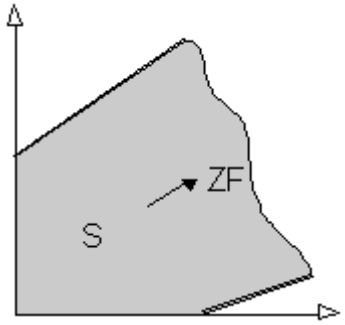
$x_1 - 3x_2 \leq 3$

	x_1	x_2	x_3	x_4	r.S.
Z	-2	-1	0	0	0
x_3	-1	1	1	0	2
x_4	1	-3	0	1	3

	x_1	x_2	x_3	x_4	r.S.
Z	0	-7	0	2	6

x_3	0	-2	1	1	5
x_1	1	-3	0	1	3

Als Pivotspalte kommt nur mehr Spalte 2 in Frage. Dort gibt es aber nur negative Einträge und daher kein Pivot möglich!

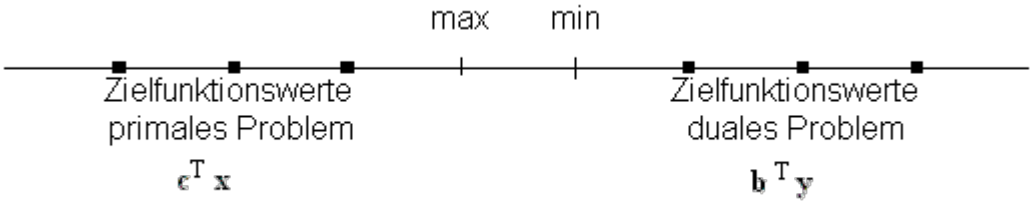


Beziehungen primales/duales Problem:

In folgender Tabelle werden alle möglichen Kombinationen angegeben:

		(D)		
		endl. Lösung	unlösbar	unbeschränkt
(P)	endl. Lösung	X		
	unlösbar		X	X
	unbeschränkt		X	

Für die Zielfunktionswerte muß ja gelten:



Daher gilt, wenn eines der beiden Probleme (primal oder dual) unbeschränkt ist, daß das andere Problem keine Lösung haben kann und daher unlösbar sein muß.
Es ist aber möglich, daß beide Probleme unlösbar sind:

Primal (P):
$x_2 \rightarrow$ max!
$x_1 - x_2 \leq -1$
$-x_1 + x_2 \leq 0$
$x_1, x_2 \geq 0$

Dual (D):
$-y_1 \rightarrow$ min!
≥ 0

$y_1 - y_2 \leq 0$
$-y_1 + y_2 \geq 1$
$y_1, y_2 \geq 0$

(P) ist unlösbar, da $x_1 - x_2$ nicht gleichzeitig ≤ -1 und ≥ 0 sein kann.
(D) ist unlösbar, da $y_1 - y_2$ nicht gleichzeitig ≥ 0 und ≤ -1 sein kann.

D. Zyklenbildung

Beispiel 2.9.:

$10\ x_1 - 57\ x_2 - 9\ x_3 - 24\ x_4 \rightarrow \max!$
$1/2\ x_1 - 11/2\ x_2 - 5/2\ x_3 + 9\ x_4 \leq 0$
$1/2\ x_1 - 3/2\ x_2 - 1/2\ x_3 + x_4 \leq 0$
$x_1 \leq 1$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	r.S.
Z	-10	57	9	24	0	0	0	0
x_5	1/2	-11/2	-5/2	9	1	0	0	0
x_6	1/2	-3/2	-1/2	1	0	1	0	0
x_7	1	0	0	0	0	0	1	1

Ecke ist degeneriert, da die Basisvariablen x_5 und x_6 den Wert 0 haben!
Wir haben 2 Zeilen als Pivotzeile zur Auswahl. Welche man auch nimmt, der nächste ZF-Wert wird wieder 0 sein (ist also zumindest nicht gesunken).
Bei ungünstiger Wahl der Pivotzeile durch den Algorithmus kann es passieren, daß man nach einigen Schritten wieder auf das selbe Tableau zurückkommt und sich ab dann immer "im Kreis dreht". Das nennt man dann **Zyklenbildung**.
Zyklenbildungg ist nur an degenerierten Ecken möglich! (Denn falls in der Pivotzeile der Wert $b_i > 0$, so steigt der ZF-Wert auf jeden Fall an!)
Das Auftreten von degenerierten Ecken ist in größeren LPs in der Praxis nicht auszuschließen. Es muß aber nicht unbedingt zu Zyklenbildung führen
(siehe etwa Beispiel 2.3, degenerierte Ecken, aber keine Zyklenbildung).
Tatsächlich ist Zyklenbildung in der Praxis äußerst unwahrscheinlich; viele Simplex-Algorithmen Implementierungen fangen sie gar nicht ab.

Durch geeignete Auswahlregeln für die Pivotzeile kann man Zyklenbildung verhindern (Lexikographische Zusatzregeln).

2.6. Parametersensitivität

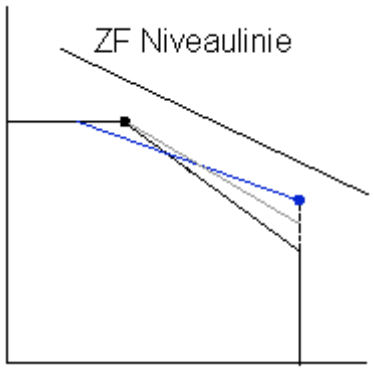
1) Sensitivität in den Nebenbedingungen

Bereits bei der Interpretation der dualen Variablen haben wir die Sensitivität in den Nebenbedingungen untersucht.
Frage: Wie weit kann man die b_i verändern (ceteris paribus), ohne daß sich die duale Lösung verändert ?

Wir hatten beim Übergang $\mathbf{b} \rightarrow \tilde{b}$ folgende ZF-Zeile:

$$-c^T + y^{*T}A \qquad y^{*T} \qquad y^{*T}\tilde{b}$$

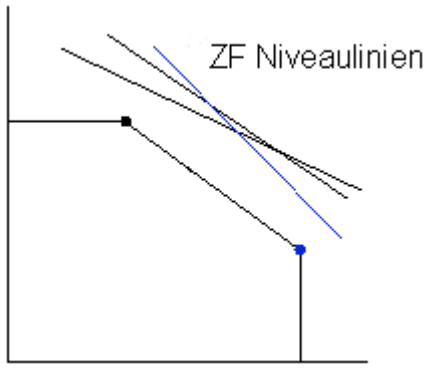
Die duale Variable soll unverändert bleiben. Stört man aber **b** zu stark, so ändert sich die duale Lösung und die primale Lösung "springt" zu einer anderen Ecke, statt sich stetig zu verändern.



Wenn man die NB zu sehr verändert, "springt" der Lösungspunkt.

2) Sensitivität in der Zielfunktion

Man kann auch die c_i verändern.
Frage: Wie weit kann man das tun, ohne daß man die primale Lösung verändert ?

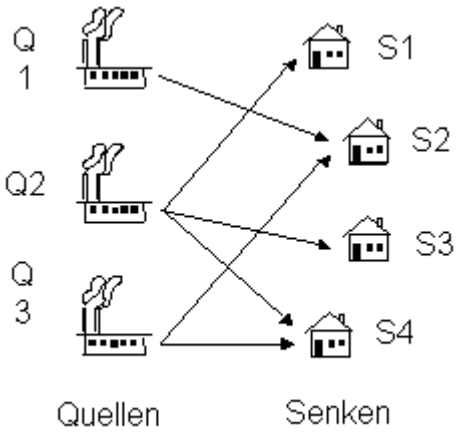


Bei kleinen Veränderungen bleiben die selben Nebenbedingungen aktiv bzw. passiv, bei größeren kommt es zu einem Austausch.

2.7. Spezielle Anwendungen der Linearen Optimierung

LPs treten in sehr vielen verschiedenen Anwendungsbereichen auf. Wir greifen Anwendungen mit speziellen formalen Strukturen heraus.

A. Transportproblem



Allgemein gibt es k Produzentenplätze (Quellen) und l Konsumentenplätze (Senken). k muß nicht gleich l sein.

c_{ij} ... Transportkosten für eine Einheit des Gutes von Quelle Q_i nach Senke S_j

a_i ... Anzahl der in Q_i produzierten Einheiten

b_i ... Anzahl der in S_j benötigten Einheiten

Wir setzen $\sum_{i=1}^{k} a_i = \sum_{i=1}^{l} b_j$ voraus, d.h. es wird insgesamt genau so viel produziert wie verbraucht.

Entscheidungsvariablen:

x_{ij} ... Menge, die von Q_i nach S_j transportiert wird ($1 \leq i \leq k$, $1 \leq j \leq l$)

Das führt uns auf folgendes LP:

$$\sum_{i=1}^{k} \sum_{j=1}^{l} c_{ij} x_{ij} \rightarrow \min!$$
$$\sum_{j=1}^{l} x_{ij} = a_i \qquad (1 \leq i \leq k)$$
$$\sum_{i=1}^{k} x_{ij} = b_j \qquad (1 \leq j \leq l)$$
$$x_{ij} \geq 0 \qquad (1 \leq i \leq k, 1 \leq j \leq l)$$

Bei näherer Betrachtung sieht man, daß das Gleichungssystem der Nebenbedingungen linear abhängig ist.
Man kann also eine Zeile der Nebenbedingungen weglassen (i.A. die letzte).

Lösungsverfahren:
a) Man bringt das LP in Standardform und wendet den Simplex Algorithmus an
oder
b) Man nützt die spezielle Struktur aus:

- Findet eine zulässige Lösung mit Hilfe der sogenannten "**Nordwesteckenregel**": beginnt mit x_{11} (also im Nordwesten der Matrix) und setzt jeweils die größtmögliche mit den

mit x_{11} (also im Nordwesten der Matrix) und setzt jeweils die größtmögliche mit den Nebenbedingungen verträgliche Transportmenge ein.
- Verbessert die Lösung mit dem **Stepping-Stone-Verfahren** (Verschiebung der transportierten Güter, so daß die Kosten sinken)

B. Zuordnungsproblem

Beispiel 2.10.: 5 Arbeiter (Q_1 ... Q_5) haben 5 Aufgaben (S_1 ... S_5) zu erledigen. Der Zeitbedarf jedes Arbeiters für jede Arbeit ist bekannt:

	S_1	S_2	S_3	S_4	S_5
Q_1	5	8	2	4	5
Q_2	4	7	1	3	4
Q_3	5	5	1	4	6
Q_4	6	7	3	4	1
Q_5	4	6	2	9	5

Frage: Welcher Arbeiter soll welche Tätigkeit verrichten, damit die Gesamtarbeitszeit minimal wird ?

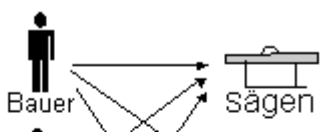
Allgemein hat man Mengen $\{Q_1,..., Q_n\}$ und $\{S_1,..., S_n\}$. Man ordne jedem Q_i (bijektiv) ein S_j zu. Kosten bei der Zuordnung von Q_i zu S_j sind c_{ij}.
Ziel: Gesamtkosten -> min!

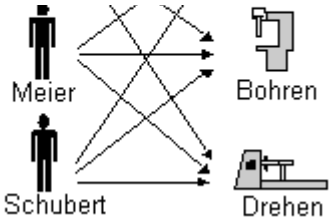
Entscheidungsvariable:
x_{ij} = 1, wenn Q_i S_j zugeordnet wird
x_{ij} = 0, ansonsten.

Eigentlich ist das ein ganzzahliges Optimierungsproblem!
Es zeigt sich jedoch, daß - auch wenn man für x_{ij} nichtganzzahlige Werte zuläßt - bei folgender Formulierung als LP nur Werte 0 oder 1 als Lösungen herauskommen.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \rightarrow \min!$$
$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (1 \le i \le n)$$
$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (1 \le j \le n)$$
$$x_{ij} \ge 0 \qquad (1 \le i \le n, 1 \le j \le n)$$

Man sieht also das Zuordnungsproblem ist ein Spezialfall des Transportproblems mit k = l = n, a_i = 1, b_i = 1.





Man kann sich vorstellen, daß von jedem Arbeiter eine "symbolische Einheit" zu der Aufgabe transportiert wird, die er übernimmt. Jede Aufgabe muß umgekehrt genau eine symbolische Einheit bekommen.

Lösungsverfahren:
Wie bei Transportproblem. Es gibt auch eine speziell auf das Zuordnungsproblem zugeschnittenes Verfahren, die sogenannte **Ungarische Methode** (von ungar. Mathematikern entwickelt).

C. Matrixspiele

Allgemeine Situation:

		Entscheidungsalternativen f. Betrieb 2			
		F1	**F2**	**.....**	**Fn**
Entscheidungs-alternativen f. Betrieb 1	**E1**	a_{11}	a_{12}	.....	a_{1n}
	E2	a_{21}	a_{22}	.....	a_{2n}
	. . .	. . .	. . .	. . .	. . .
	Em	a_{m1}	a_{m2}	.....	a_{mn}

Nullsummenspiel:
a_{ij} : Auszahlung f. Betrieb 1 bei (i, j)
$-a_{ij}$: Auszahlung f. Betrieb 2 bei (i, j)
Summe der Auszahlungen f. Betrieb 1 und 2 ist gleich 0.
Betrieb 1 und 2 sind vollständige Kontrahenten (was einer gewinnt, verliert der andere)

Spiel ist durch Auszahlungsmatrix **A** = (a_{ij}) gegeben, daher der Name Matrixspiel.
Auch bei Sherlock Holmes gegen Dr. Moriarty liegt ein Matrixspiel vor.
Auszahlungsmatrix ist $\begin{pmatrix} 0 & 1/2 \\ 1 & 0 \end{pmatrix}$
Eine Lösung: Minimax-Strategie. Diese beruht auf der (pessimistischen) Voraussetzung, daß der Gegner meine Entscheidung "vorhersieht".
D.h.: Wähle ich (=Zeilenspieler) die Zeile i, so wählt er (=Spaltenspieler) jene Zeile j, die a_{ij} minimiert. Also muß ich i so wählen, daß $\min_j a_{ij}$ maximiert wird.
Im Sherlock Holmes Beispiel ist das $\min_j a_{ij}$ jeweils gleich 0 und damit das Maximum davon auch gleich 0. Ich komme über die Auszahlung 0 nicht hinaus.

Retten kann ich mich nur, wenn ich einen Mechanismus verwende, dessen Ausgang mein Gegner deshalb nicht vorhersehen kann, weil ich ihn selber nicht weiß - einen Zufallsmechanismus.
Das führe auf "gemischte Strategien":

Gemischte Strategie:

p_i = Wahrsch. f. Entscheidung Ei (i = 1, ..., m)

$p_1 + p_2 + ... + p_m = 1$

Erwartungswert der Auszahlung, falls Gegner reine Strategie Fj spielt:

$p_1 a_{1j} + p_2 a_{2j} + ... + p_m a_{mj}$

Ich möchte mir - egal was der Gegner tut - eine erwartete Auszahlung von w sichern. Es muß dann für jedes j

$p_1 a_{1j} + p_2 a_{2j} + ... + p_m a_{mj} \geq w$ sein.

Damit bekommt man folgendes LP:

$w \rightarrow$ max!

$p_1 a_{1j} + p_2 a_{2j} + ... + p_m a_{mj} - w \geq 0 \quad (1 \leq j \leq n)$

$p_1 + p_2 + ... + p_m = 1$

$p_i \geq 0 \quad (1 \leq i \leq n)$

Entscheidungsvariablen: w, p_1, p_2, ... , p_m

Lösung: Simplex Algorithmus. Die duale Lösung bildet die optimale gemischte Strategie des Gegners.

$$\left\{\begin{array}{lcl} w & \rightarrow & \max! \\ w & \leq & p_2 \\ w & \leq & \frac{1}{2}p_1 \\ p_1 + p_2 & = & 1 \\ w, p_1, p_2 & \geq & 0 \end{array}\right. \qquad (P)$$

$$\left\{\begin{array}{lcl} z & \rightarrow & \min! \\ z & \geq & \frac{1}{2}q_2 \\ z & \geq & q_1 \\ q_1 + q_2 & = & 1 \\ z, q_1, q_2 & \geq & 0 \end{array}\right. \qquad (D)$$

3. Nichtlineare Optimierung

3.1. Einleitung und Klassifikation

Bei Linearer Optimierung sind Zielfunktion und Nebenbedingungen linear!

Im Folgenden behandeln wir den Fall, daß entweder Zielfunktion oder Nebenbedingungen nicht linear sind (oder beides). Dafür benötigt man eigene Algorithmen.

Verschiedene Möglichkeiten die Probleme einzuteilen:

- Konvex / nicht konvex
- Mit / ohne Nebenbedingungen
- Lokale / Globale Optimierung
- Deterministische / Stochastische Optimierung

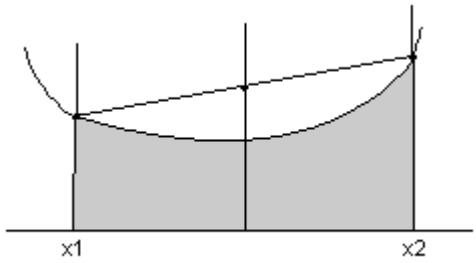
Wie bei der linearen, so spricht man auch bei der nichtlinearen Optimierung oft von "Programmierung" statt von "Optimierung".

Lösungs-Algorithmen der nichtlinearen Optimierung ist im Prinzip auch auf LPs anwendbar, der Simplex Algorithmus ist aber bei LPs effizienter.

3.2. Konvexe Optimierung

Definition: Eine Funktion f(x) heißt **konvex**, wenn für alle x_1, x_2 und λ ($0 \leq \lambda \leq 1$) gilt:
$f(\lambda.x_1 + (1 - \lambda).x_2) \leq \lambda f(x_1) + (1 - \lambda) f(x_2)$

Gilt die Gleichheit, so ist die Funktion linear.

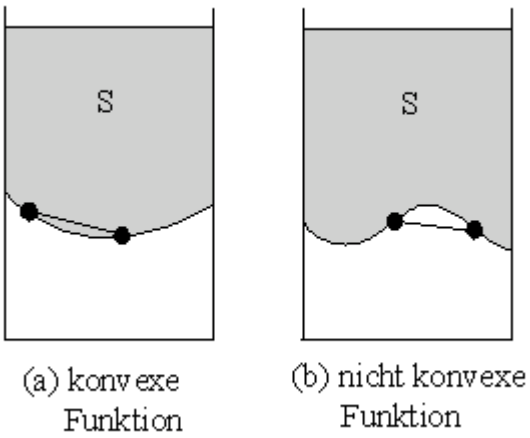


Der Funktions-Graph liegt immer **unter** der Verbindungsgeraden zweier Punkte!

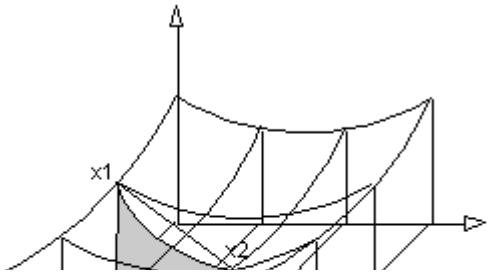
Definition: Eine Funktion f(x) heißt **konkav**, wenn für alle x_1, x_2 und λ ($0 \leq \lambda \leq 1$) gilt:
$f(\lambda.x_1 + (1 - \lambda).x_2) \geq \lambda f(x_1) + (1 - \lambda) f(x_2)$
("$\leq$ " durch "$\geq$" ersetzt)

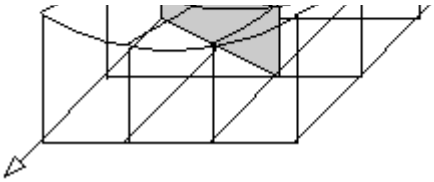
Falls f(x) 2-mal differenzierbar ist, so gilt:
f(x) ist konvex, falls f"(x) $\geq$ 0 für alle x
f(x) ist konkav, falls f"(x) $\leq$ 0 für alle x

Zusammenhang konvexe Funktion / konvexe Menge:
Eine Funktion ist konvex, falls die Menge S der Punkte "oberhalb" des Funktionsgraphen eine konvexe Menge ist.



Die Definitionen gelten auch für Funktionen in mehreren Variablen:





Definition: Unter einem **konvexen Optimierungsproblem** versteht man ein Optimierungsproblem, der Form

$f(x) \rightarrow \min!$
$g_1(x) \leq b_1$
.
.
$g_m(x) \leq b_m$
mit konvexen Funktionen f, g_1 , ..., g_m

3.3. Optimierung mit / ohne Nebenbedingungen

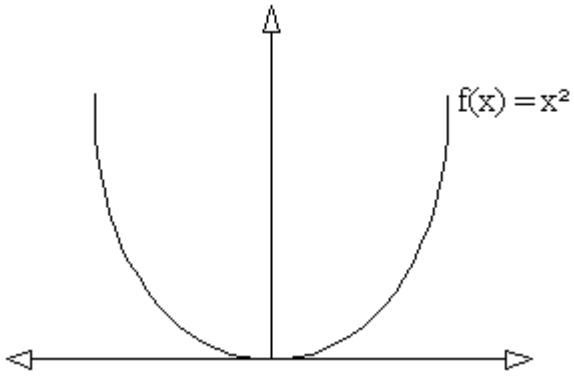
Englisch: constrained / unconstrained optimization

$f(x) \rightarrow \min!$
$x \in \mathbf{S}$

Ohne Nebenbedingungen: $\mathbf{S} = \mathbf{R}^n$ (im linearen Fall ist das nicht sinnvoll)

Beispiel: *Optimierung ohne Nebenbedingungen*

$x^2 \rightarrow \min!, x \in \mathbf{R}^1$



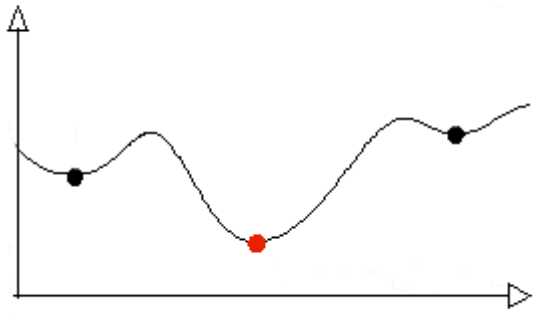
3.4. Lokale / globale Optimierung

Englisch: local / global optimization

Ein lokales Optimum ist eine Stelle, in deren "näherer Umgebung" es keine Stelle mit einem besseren Zielfunktions-Wert gibt.

Ein globales Optimum ist eine Stelle, für die es nirgends in **S** eine Stelle mit einem besseren Zielfunktions-Wert gibt.

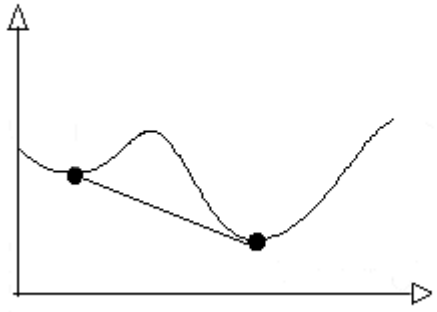
Beispiel:



Bei **strikt konvexen** ($f(\lambda.x_1 + (1 - \lambda).x_2) < \lambda\, f(x_1) + (1 - \lambda)\, f(x_2),\ \ 0 < \lambda < 1$)
Optimierungsproblemen ohne Nebenbedingungen kann es nur **ein** lokales Minimum geben, dieses ist also gleichzeitig globales Minimum.

Also ist jedes lokale Optimum schon globales Optimum!

Denn angenommen es gäbe 2 lokale Minima:



dann könnte man die beiden durch eine Strecke verbinden und der Funktions-Graph müßte oberhalb statt unterhalb der Strecke liegen!

Diese Eigenschaft macht konvexe Optimierungsprobleme so "angenehm".

3.5. Deterministische / Stochastische Optimierung

Englisch: deterministic / stochastic optimization

Deterministische Optimierung:

- Zielfunktion und Nebenbedingungen fest vorgegeben
- Lösungsalgorithmus ist auch deterministisch

Stochastische Optimierung:

1) **Zufallseinfluß auf Zielfunktion und/oder Nebenbedingungen**:

Beispiel:
Verarbeitung von Mineralien. Die Gestehungspreise für die Rohstoffe R1, R2 können - nachdem man die Bestellung aufgegeben hat - noch um bis zu 10% schwanken.

Einfachstes Lösungsverfahren: **"Szenarienanalyse"**

Szenario 1: Preise sinken um 10%, also: R1 = 225, R2 = 180, Wahrscheinlichkeit p1= 0.1
Szenario 2: Preise bleiben gleich, also: R1 = 250, R2 = 200, Wahrscheinlichkeit p2= 0.5
Szenario 1: Preise steigen um 10%, also: R1 = 275, R2 = 220, Wahrscheinlichkeit p3= 0.4

Neue Zielfunktion = p1.ZF1 + p2.ZF2 + p3.ZF3 = $0.1(225y_1 + 180y_2) + 0.5(250y_1 + 200y_2) + 0.4(275y_1 + 220y_2) = 257.5y_1 + 206y_2$

Auch Schwankungen in den Nebenbedingungen lassen sich auf diese Weise behandeln.
Will man nicht nur einfach eine endliche Anzahl von Szenarien unterscheiden, sondern z.B. einen Preis verteilt nach N(Mittelwert, Standardabw.), so wird es mathematisch aufwendiger.

2) **Lösungsalgorithmus stochastisch** ("Künstliches" Ins-Spiel-Bringen des Zufalls):

Vor allem zu dem Zweck, um von lokalen Optima wieder wegzukommen und - hoffentlich - irgendwann das globale Optimum zu finden.

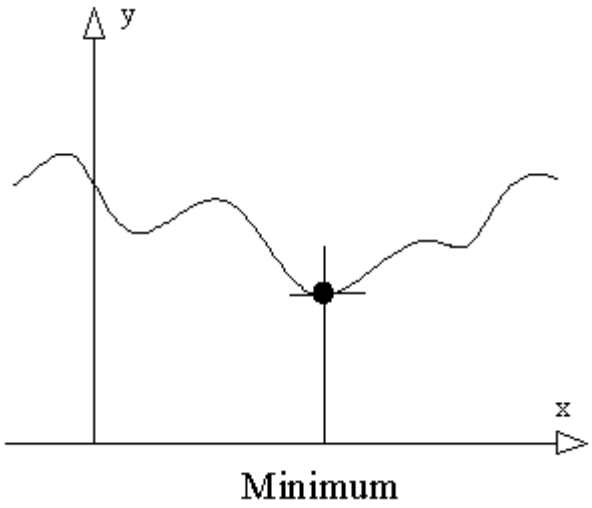
Beispiel:
Ein Wanderer, der stets in Richtung des steilsten Anstieges geht, landet auf dem nächsten Berggipfel (= lokales Maximum) und bleibt dort eventuell sitzen.

Ein stochastisches Verfahren wie etwa "Simulated Annealing" geht zwar eher bergauf, aber manchmal (zufällig) auch wieder bergab.
Zu Beginn werden die Schritte "nach oben" und "nach unten" eher zufällig sein. Je länger der Algorithmus allerdings läuft, deste stärker wird die Neigung "nach oben" zu gehen. Der Simulated Annealing Algorithmus konvergiert gegen das globale Optimum. Bei endlicher Laufzeit, werden in der Praxis allerdings unter Umständen nur lokale Optima gefunden.

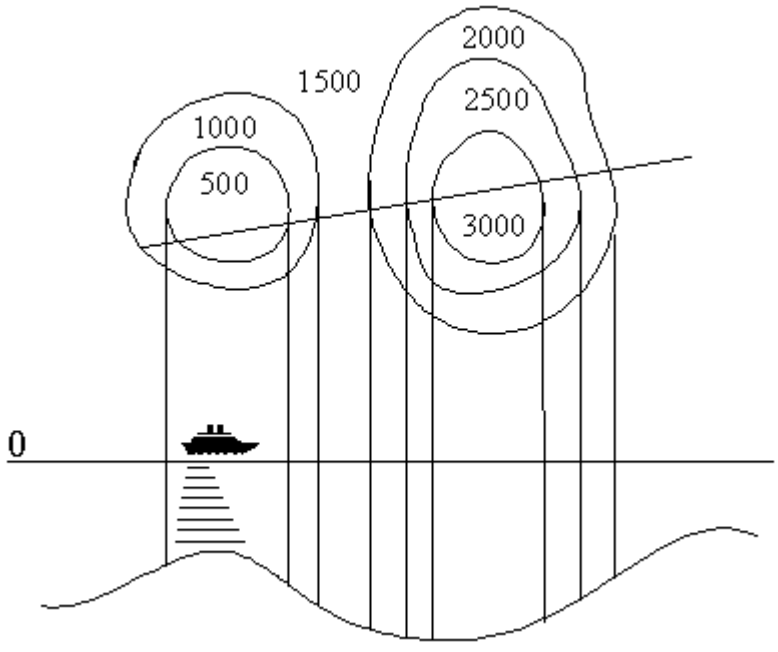
4. Nichtlineare Optimierung ohne Nebenbedingungen

Zielfunktion **F**: $\mathbf{R}^n \rightarrow \mathbf{R}$
F(x) $\rightarrow$ min!

4.1. Eindimensionale Optimierung



Diese Situation tritt auch im mehrdimensionalen Fall auf, wenn das Minimum entlang einer Geraden gesucht wird ("**line search**"), z.B. wird etwa entlang einer bestimmten Richtung die tiefste Stelle im Meer gesucht.



Das führt im mehrdimensionalen Fall natürlich noch nicht direkt zum (globalen) Minimum, wird aber als Hilfsroutine für Minimierungsverfahren häufig eingesetzt.

Überblick über die Verfahren:

A) Elementare Line-Search Verfahren (ableitungsfrei)
1) Äquidistante Suchpunkte

1) Äquidistante Suchpunkte
2) Zufallssuche (Random Search)
3) Fibonacci Suche
4) Methode des Goldenen Schnitts

B) Suchverfahren mit Ableitung

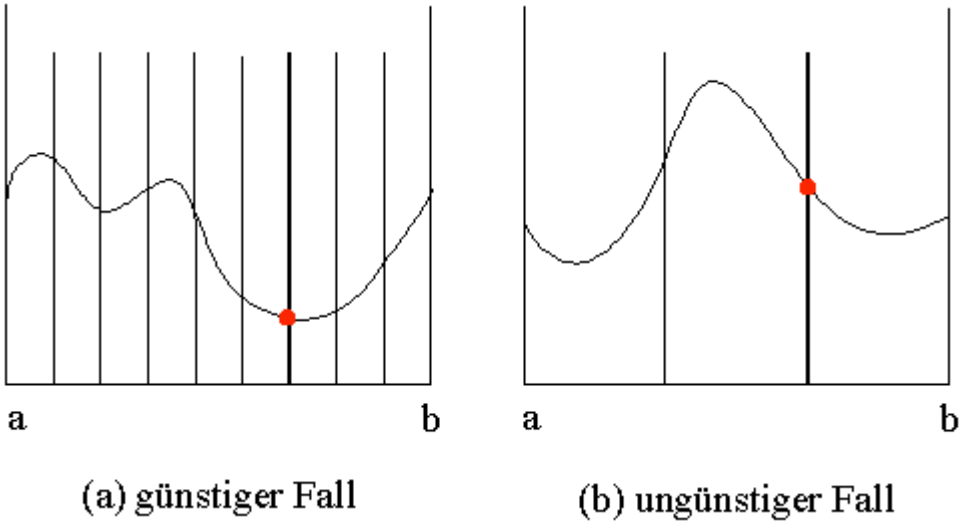
5) Das eindimensionale Newton-Verfahren

A) Elementare Line-Search Verfahren

Es werden nur Funktionswerte berechnet. Die Ableitung der Funktion ist nicht notwendig.
Einschränkung auf hinreichend großes Intervall.
Eigentlich handelt es sich bei den nachfolgenden um Probleme mit Nebenbedingungen (es gilt $a \leq x \leq b$).
Im allgemeinen kann aber ein Intervall angegeben werden, in dem das Minimum liegen muß.

1) Äquidistante Suchpunkte

Funktionswerte werden in gleichen Abständen bestimmt. Man nimmt den Punkt mit dem kleinsten Funktionswert.

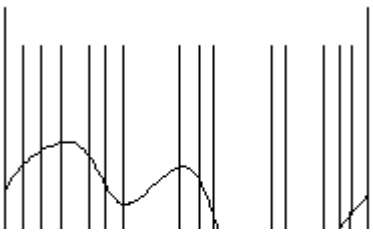


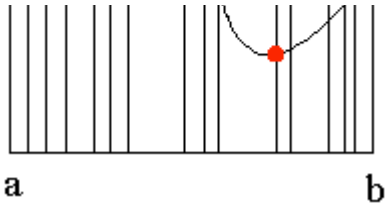
(a) günstiger Fall (b) ungünstiger Fall

Im Fall (a) erreichen wir ein recht gutes Ergebnis. Im Fall (b) wurden die Suchpunkte aber ungünstig gewählt und so wird ein x-Wert gefunden, der relativ weit vom wirklichen Minimum entfernt liegt.

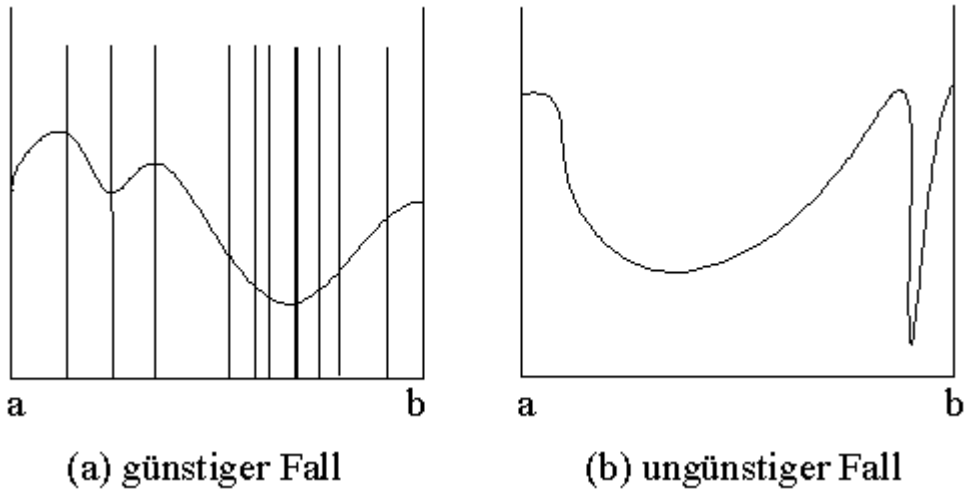
2) Zufallssuche (Random Search)

(Pseudo) Zufallszahlen zwischen a und b werden als Suchpunkte, an denen Funktionswerte bestimmt werden, gewählt.
Im eindimensionalen Fall ist das eher ungünstiger. Erst im mehrdimensionalen Fall ist die Verwendung von Random-Search-Algorithmen günstiger.



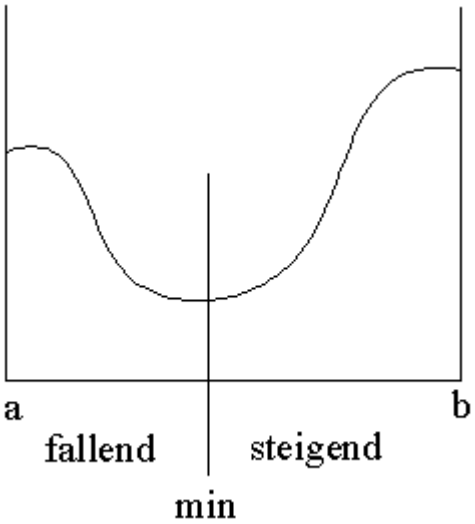


Wenn man nichts über die Eigenschaften der Zielfunktion weiß, kann man auch nicht sehr viel mehr machen, als eines der beiden obigen Verfahren zu verwenden.
D.h. man probiert einfach "irgendwelche" Werte aus.
Adaptive Suchverfahren versuchen Informationen aus den vorhergegangenen Suchschritten auszunützen. Bei relativ glattem Verlauf der Zielfunktion wird das
globale Optimum gefunden. In ungünstigen Fällen wird nur ein lokales Optimum gefunden.



Zielfunktionen in wirtschaftlichen Anwendungen haben aber meistens eine "schönere" Struktur und gewisse Eigenschaften, die wir im voraus wissen können.

Wir setzen daher im Folgenden "**unimodale**" Zielfunktionen voraus, d.h. daß sie im untersuchten Intervall genau ein lokales Optimum haben.

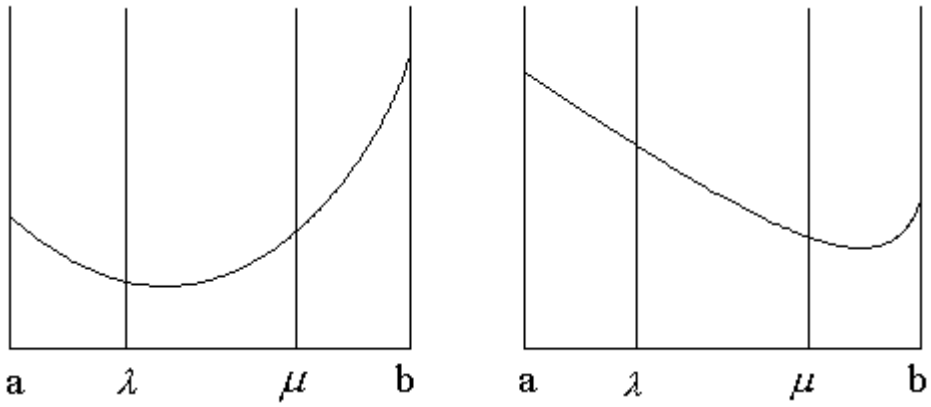


Insbesondere ist jede strikt konvexe Funktion innerhalb jedes Intervalls [a,b] unimodal, da sie dort ein lokales Minimum haben muß (evtl. am Rand) und auch nicht mehr als ein lokales Minimum

haben kann.
Wir wollen also nun "adaptiv" vorgehen, d.h. die jeweils nächsten Suchpunkte sind abhängig von den Funktionswerten der vorhergehenden Suchpunkte (man konzentriert sich auf Punkte, die sich "lohnen"). Allgemein ist das Prinzip so, daß das Intervall das das Minimum enthält von Schritt zu Schritt weiter eingeengt wird.

Voraussetzung für die Ausnutzung der unimodalen Funktionsform:

- [a,b] enthält das Minimum von F(x)
- man kennt F(x) in Punkten λ und μ, wobei $a < \lambda < \mu < b$.



Das Intervall [a,b] kann nun folgendermaßen eingeengt werden:

a) Falls $F(\lambda) < F(\mu)$: Das Optimum kann nicht in $[\mu, b]$ liegen $\rightarrow$ neues Intervall $[a, \mu]$
b) Falls $F(\lambda) > F(\mu)$: Das Optimum kann nicht in $[a, \lambda]$ liegen $\rightarrow$ neues Intervall $[\lambda, b]$

Mit dem "verkürzten" Intervall $[a, \mu]$ oder $[\lambda, b]$ im nächsten Schritt analog weitermachen.

Wie wählt man nun λ und μ?

Günstig wäre es, wenn man z.B. bei Einengung von [a, b] auf $[a, \mu]$ den Wert λ als nächsten Unterteilungspunkt gleich beibehalten könnte. Dann muß man pro Schritt nur einen zusätzlichen Funktionswert berechnen (die Funktionsberechnung kann ja unter Umständen extrem aufwändig sein, z.B. wenn erst durch eine Simulationsroutine der Zielfunktionswert bestimmt wird).

3) Fibonacci-Suche

Fibonacci-Zahlen sind rekursiv definiert:
$c_0 = c_1 = 1$
$c_k = c_{k-1} + c_{k-2}$

Beispiel: *(Hasen züchten)*
Man beginnt im Jahr 0 mit einem jungen Hasenpaar (h: junges Hasenpaar, H: erwachsenes Hasenpaar)

Jahr	Hasen	Fibonacci-Zahlen
0	h	1

1	H	1
2	Hh	2
3	HHh	3
4	HHHhh	5
5	HHHHHhhh	8
6	..	13
7	..	21
8	..	34
9	..	55
10	..	89

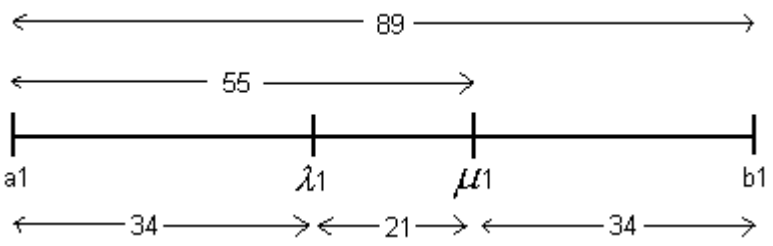
Algorithmus: ("*Fibonacci*" - *Suche*)

a_1 := a, b_1 := b ;
Für k:=1 bis n-1:
$\lambda_k := a_k + (b_k - a_k)(c_{n-k-1}/c_{n-k+1})$;
$\mu_k := a_k + (b_k - a_k)(c_{n-k}/c_{n-k+1})$;
    wenn $F(\lambda_k) < F(\mu_k)$
    dann $b_{k+1} := \mu_k$ (neues Intervall: $[a_k, \mu_k]$)
    sonst $a_{k+1} := \lambda_k$ (neues Intervall: $[\lambda_k, b_k]$)

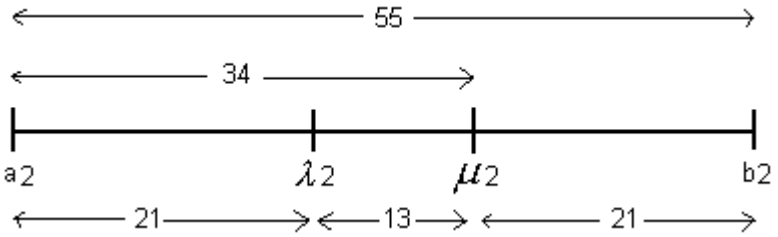
Der Algorithmus liefert eine kontrahierende Folge von Intervallen, die das Minimum enthalten. Wenn n hinreichend groß gewählt wird, kann das Intervall beliebig klein gemacht werden.

Beispiel:
$F(x) = (x - 30)^2$, a = 0, b = 89, n = 10.

1.Schritt:
k = 1
$\lambda_1 = a_1 + (b_1 - a_1)(c_{10-1-1}/c_{10-1+1}) = 89.34/89 = 34$ $F(\lambda_1) = 16$
$\mu_1 = a_1 + (b_1 - a_1)(c_{10-1}/c_{10-1+1}) = 89.55/89 = 55$ $F(\mu_1) = 625$
Also $F(\lambda_1) < F(\mu_1)$ und daher neues Intervall [0, 55] (d.h. $a_2 = 0$ und $b_2 = 55$)



2.Schritt:
k = 2
$\lambda_2 = a_2 + (b_2 - a_2)(c_{10-2-1}/c_{10-2+1}) = 55.21/55 = 21$ $F(\lambda_2) = 81$
$\mu_2 = a_2 + (b_2 - a_2)(c_{10-2}/c_{10-2+1}) = 55.34/55 = 34$ $F(\mu_2) = 16$
Also $F(\lambda_1) > F(\mu_1)$ und daher neues Intervall [21, 55] (d.h. $a_3 = 21$ und $b_3 = 55$)

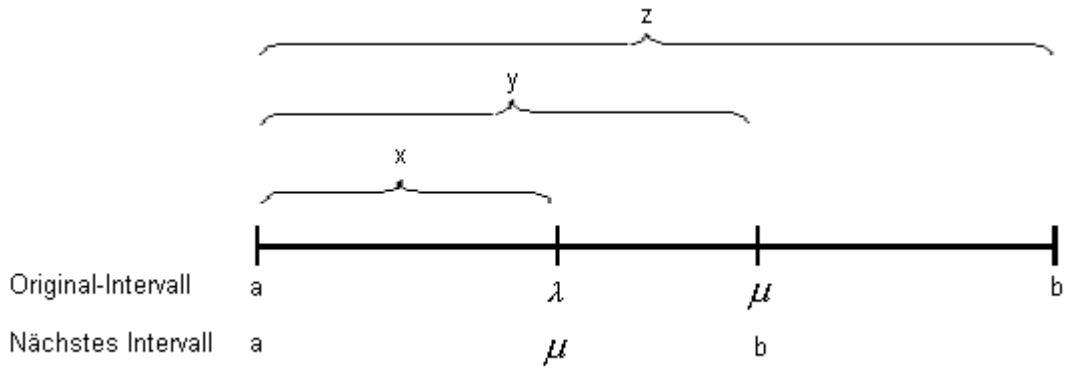


Man sieht, daß jeder "alte" Teilungspunkt "paßt", um auch als einer der beiden "neuen" Teilungspunkte verwendet werden zu können.

Nachteil: Das Teilungsverhältnis ändert sich von Schritt zu Schritt und muß immer neu berechnet werden (man muß die Fibonacci-Zahlen gespeichert haben)

4) Methode des Goldenen Schnittes

Man versucht nun ein konstantes Teilungsverhältnis zu erhalten und trotzdem nur eine Funktionsberechnung pro Schritt machen zu müssen:
Sei $\mu = a + \gamma(b-a)$ und $\lambda = a + (1-\gamma)(b-a)$ (also $\gamma = (\mu-a)/(b-a)$, konstantes Teilungsverhältnis)
Damit wir nun λ wiederverwenden können, muß gelten:
Verhältnis von $x = \lambda - a$ zu $y = \mu - a$ muß gleich sein dem Verhältnis von $y = \mu - a$ zu $z = b - a$.
Also $x : y = y : z$,
bzw. $(1-\gamma) : \gamma = \gamma \rightarrow \gamma^2 + \gamma - 1 = 0$
also ist $\gamma = \frac{\sqrt{5}-1}{2} = 0.618034....$ (das ist das Verhältnis des "Goldenen Schnittes")



Algorithmus: ("*Goldener Schnitt*" *- Suche*)

$\gamma := \frac{\sqrt{5}-1}{2}$;

```
 bis das Intervall klein genug ist:
λ := a + (1 - γ)(b-a);
μ := a + γ(b-a);
    wenn F(λ) < F(μ)
    dann b := μ    (a bleibt gleich)
    sonst a:= λ     (b bleibt gleich)
```

Der Algorithmus liefert eine kontrahierende Folgen von Intervallen [a, b], die das Minimum enthalten.

Beispiel:

$F(x) = (x - 30)^2$, a = 0, b = 100.

1.Schritt:

$\lambda = a + (1-\gamma)(b-a) = (1-\gamma).100 = 38,2$ $F(\lambda) = 67,19$

$\mu = a + \gamma(b-a) = \gamma.100 = 61,8$ $F(\mu) = 1383,17$

Also $F(\lambda) < F(\mu)$ und daher b := 61,8

2.Schritt:

$\lambda = a + (1-\gamma)(b-a) = (1-\gamma).61,8 = 23,61$ $F(\lambda) = 40,89$

$\mu = a + \gamma(b-a) = \gamma.61,8 = 38,2$ $F(\mu) = 67,19$

Also $F(\lambda) < F(\mu)$ und daher b := 38,2

3.Schritt:

$\lambda = a + (1-\gamma)(b-a) = (1-\gamma).38,2 = 14,59$ $F(\lambda) = 237,43$

$\mu = a + \gamma(b-a) = \gamma.38,2 = 23,61$ $F(\mu) = 40,89$

Also $F(\lambda) > F(\mu)$ und daher a := 14,59

4.Schritt:

$\lambda = a + (1-\gamma)(b-a) = 14,59 + (1-\gamma).(38,2-14,59) = 23,61$ $F(\lambda) = 40,89$

$\mu = a + \gamma(b-a) = 14,59 + \gamma.(38,2-14,59) = 29,18$ $F(\mu) = 0,67$

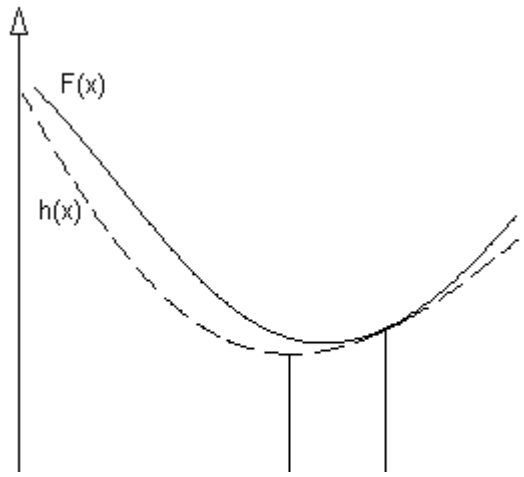
Also $F(\lambda) > F(\mu)$ und daher a := 23,61

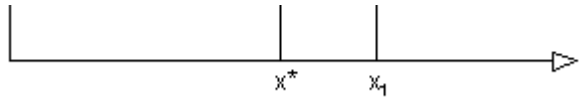
Das Minimum ist nun bereits auf das Intervall [23,61; 38,2] eingeschränkt. Man kann nun soweit weiterrechnen bis man die gewünschte Genauigkeit erreicht hat.
Das Verfahren konvergiert langsam, aber es konvergiert!

Die vorhergehenden Methoden verwendeten nur Berechnungen von Funktionswerten, nicht aber Ableitungen.
Das wichtigste Verfahren, das auch Ableitungen verwendet, ist das Newton-Verfahren.

5) Das eindimensionale Newton-Verfahren

Die Funktion F(x) wird an einer Stelle x_1 quadratisch (Polynom 2. Grades) approximiert. Man sucht das Minimum des approximierenden Polynoms und macht an dieser Stelle weiter.





Für die Approximation braucht man $F(x_1)$, $F'(x_1)$, $F''(x_1)$

Das approximierende Polynom sei h(x):
$h(x) = a + bx + cx^2$
$h'(x) = b + 2cx$
$h''(x) = 2c$

Das Minimum von h(x) erhält man:
$h'(x^*) = b + 2cx^* = 0 \rightarrow \quad x^* = - b/(2c)$
(Voraussetzung: $c > 0$, da man sonst Maximum oder Wendepunkt hätte)

Nun bestimmen wir a, b und c (tatsächlich braucht man zur Berechnung von x* nur b und c).
$h(x_1) = F(x_1)$, $h'(x_1) = F'(x_1)$, $h''(x_1) = F''(x_1)$, also
$a + bx_1 + cx_1^2 = F(x_1)$
$b + 2cx_1 = F'(x_1)$
$2c = F''(x_1)$

Daraus folgt: $c = F''(x_1)/2$ (da $c > 0$ sein muß, muß also auch $F''(x_1) > 0$ sein. Ist bei einer konvexen Funktion sicher der Fall)
$b = -2cx_1 + F'(x_1) = - x_1 F''(x_1) + F'(x_1)$

Also $x^* = - b/(2c) = - (- x_1 F''(x_1) + F'(x_1))/ F''(x_1) = (x_1 F''(x_1) - F'(x_1))/ F''(x_1) = x_1 - F'(x_1) / F''(x_1)$

Das kann man nun **rekursiv** fortsetzen:

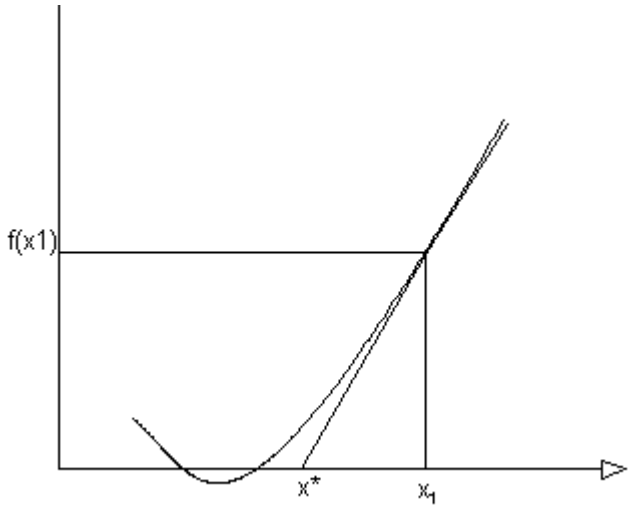
$x_{k+1} := x_k - F'(x_k) / F''(x_k)$

Algorithmus: ("*Newton-Raphson*")

```
Wähle ein geeignetes x1;
Für k = 1, 2, 3, ...., bis |xk+1 - xk| < ε
xk+1 := xk - F'(xk)/F''(xk)
```

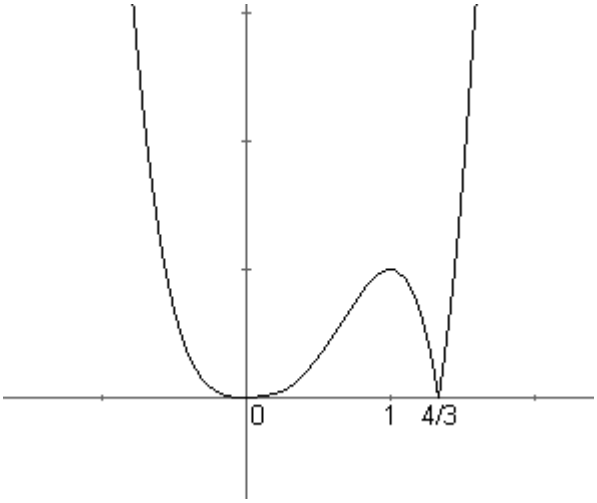
(**ε** entspricht der gewünschten Rechengenauigkeit)

Dieser Algorithmus ist verwandt mit dem Newton-Verfahren zur Nullstellenbestimmung (Lösung von Gleichungen):
Man setzt dann F' = f
Das Minimum von F = Nullstelle von f
Beim Newton-Verfahren zur Nullstellenberechnung geht man wie folgt vor:



Man legt die Tangente an die Funktion f an und schneidet sie mit der x-Achse. Dieser Schnittpunkt wird dann für den nächsten Schritt verwendet.
Also $f(x_1)/(x_1 - x^*) = f'(x_1)$ und damit $x^* = x_1 - f(x_1)/f'(x_1)$.
Das führt zu der Rekursion
$x_{k+1} = x_k - f(x_k)/f'(x_{1k})$
und das ist genau dasselbe wie oben.

Beispiel:
$F(x) = |4x^3 - 3x^4|$
$F'(x) = |12x^2 - 12x^3|$
$F''(x) = |24x - 36x^2|$



Lokales Minimum bei x = 0.
Das Newton-Verfahren liefert (bei Startwert x1 = 0.4):

k	**x_k**
1	0.4
2	0.1

3	0.047059
4	0.022934
5	0.011331
6	0.005634
.....	.....

Konvergiert bei gutem Startwert recht rasch!

Wesentlich für eine gute Konvergenz ist, daß der Startwert schon relativ nahe beim Optimum ist. Bei ungünstiger Wahl kann es auch passieren, daß sich der iterierte Wert vom Optimum weg bewegt.

Beim Newton-Raphson Verfahren erhält man keine Aussage darüber, ob das Optimum global oder lokal ist.

4.2. Mehrdimensionale Optimierung

Wir betrachten jetzt Funktionen **F**: $\mathbf{R}^n \rightarrow \mathbf{R}$ mit $n \geq 2$, die wir minimieren wollen

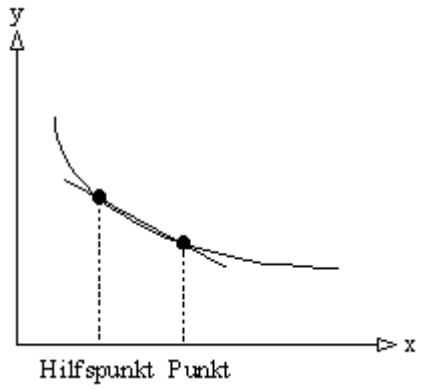
Also: $F(x) \rightarrow \min!$ mit $x = \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$

Wie schon im 1-dimensionalen Fall, so kann man auch hier zwischen Methoden unterscheiden, die nur Funktionswerte verwenden, uns solche bei denen auch die Kenntnis von (ersten oder zweiten) Ableitungen vorausgesetzt wird.

Zur Ermittlung der Funktionswerte gibt es folgende Möglichkeiten:

- Direkte Beobachtung
- Experiment (z.B. Variation von Parametern im Produktionsprozeß)
- Simulation (z.B. Simulation von Ampelphasen)
- Numerische Routinen
- Mathematische Formeln (in der Praxis eher selten)

Nur im letzten Fall kann man i.A. die Ableitungen direkt berechnen. Ansonsten ist man auf Schätzungen angewiesen.
Wenn Funktionswerte bestimmt werden können, kann man auch den Gradienten schätzen. Man geht dazu so vor, daß man zu einem gegebenen Punkt den Funktionswert bestimmt und auch für einen zweiten Punkt "in der Nähe" (liegt er zu nahe, kommt es zu starken Schwankungen, liegt er zu weit entfernt, hat der geschätzte Wert keine Aussagekraft). Die Steigung der Verbindungsgeraden wird dann als Schätzung verwendet werden.



Überblick über die folgenden Verfahren:

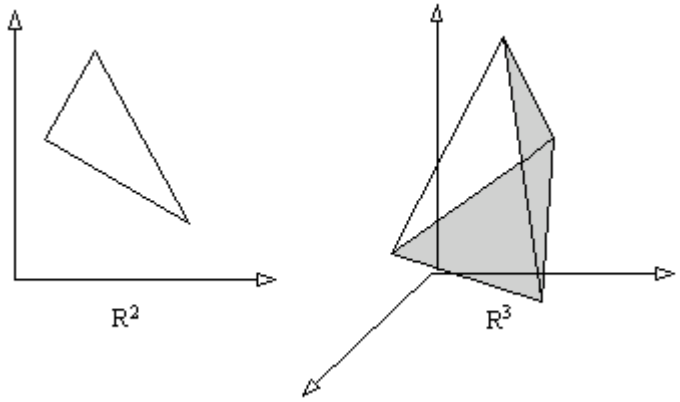
- Verfahren ohne Verwendung der Ableitung
  - Polytop-Methode (Nelder & Mead)
- Verfahren mit Verwendung der Ableitung
  - Methode des steilsten Abstiegs (Steepest descent)
  - Das mehrdimensionale Newtonverfahren

Polytop-Methode (Nelder & Mead)

Idee: Bestimme F and den Eckpunkten eines Simplex (für n = 2 ist das ein Dreieck) und adaptiere

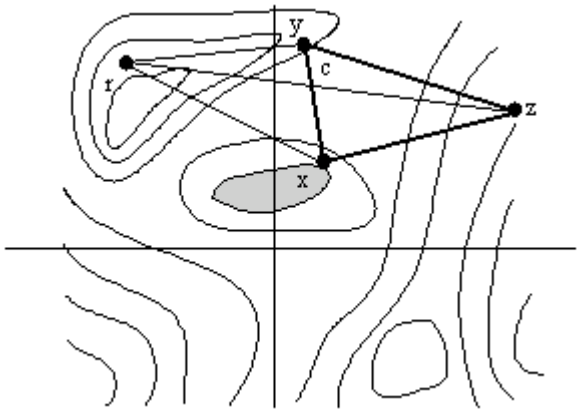
Idee: Bestimme F and den Eckpunkten eines Simplex (für n = 2 ist das ein Dreieck) und adaptiere den Simplex dann so, daß man sich besseren Werten nähert.

Ein Simplex im $\mathbf{R}^n$ ist ein Polyeder mit genau n+1 Eckpunkten.



Algorithmus für n = 2 (für n > 2 analog):

1) Beginne mit (irgendeinem) Startsimplex
2) Im k-ten Schritt: Die Eckpunkte des Simplex x, y, z werden so geordnet, daß $F(x) \leq F(y) \leq F(z)$



Für z (Punkt mit höchstem Funktionswert) wird durch "Umklappen" des Dreiecks ein neuer möglicher Wert bestimmt:

$c := (x + y) / 2$

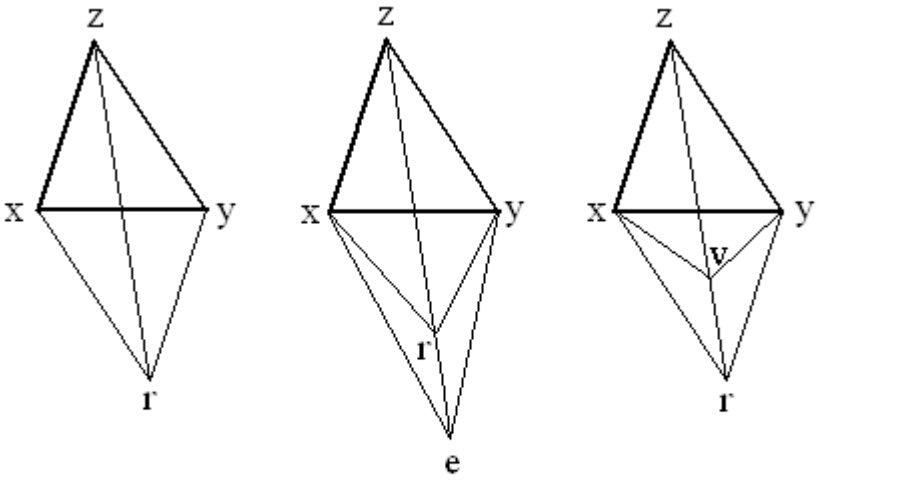
$r := c + \alpha\,(c - z)$ mit $\alpha > 0$

Für den neuen Punkt r gibt es 3 mögliche Fälle:

a) $F(x) \leq F(r) \leq F(y)$
z := r und man setzt mit dem "geklappten" Dreieck fort (y ist der neue "schlechteste" Punkt)

b) $F(r) < F(x)$
Sehr gute Wahl, daher wird das "geklappte" Dreieck noch erweitert
$e := c + \beta\,(r-c)$ mit $\beta > 1$
wenn $F(e) < F(r)$, dann wird z durch e ersetzt
wenn $F(e) \geq F(r)$, dann wird z durch r ersetzt

c) $F(r) > F(y)$
der neue Punkt r ist wieder der "schlechteste". Wir befinden uns bereits in der Talsohle, es wird daher das Dreieck verkleinert, damit es sich um das Minimum herum zusammenziehen kann.
$v := c + \gamma\,(r - c)$ mit $\gamma < 1$
z wird durch v ersetzt.



(a) neues Dreieck verwenden

(b) erweitertes Dreieck

(c) verkleinertes Dreieck

Verfahren mit Verwendung der Ableitung

F: $R^n \rightarrow R$

Sei $\mathbf{d} \in R^n$, $\| \mathbf{d} \| = \sqrt{d_1^2 + \ldots + x_n^2} = 1$ ein normierter Richtungsvektor.

Mehrdimensionaler Tayler'scher Lehrsatz: $F(\mathbf{x} + \lambda \mathbf{d}) = F(\mathbf{x}) + \lambda\, \mathbf{d} \nabla F(\mathbf{x}) \mathbf{d} + \frac{1}{2}\lambda^2\, \mathbf{d}' \mathbf{H}_F(\mathbf{x}) \mathbf{d} + \ldots$

wobei $\nabla F = \left(\frac{\partial F}{\partial x_1}, \ldots, \frac{\partial F}{\partial x_n} \right)$ der **Gradient**

und $H_F = \left(\frac{\partial^2 F}{\partial x_i x_j} \right)_{1 \le i \le n, 1 \le j \le n}$ die **Hessche'sche Matrix** (Matrix der 2. partiellen Ableitungen, eine symmetrische Matrix, da $\frac{\partial^2 F}{\partial x_i x_j} = \frac{\partial^2 F}{\partial x_j x_i}$)

Theorem:

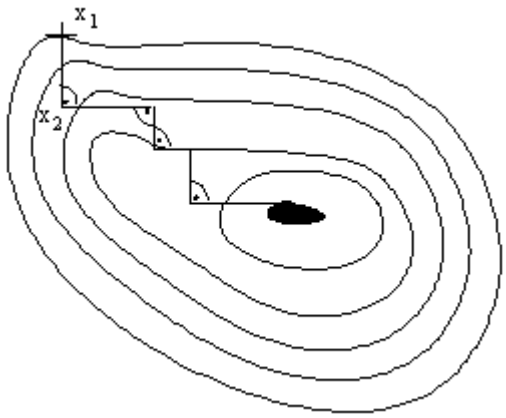
∇F zeigt immer in Richtung des **steilsten Anstieg**s und $- \nabla F$ zeigt immer in Richtung des **steilsten Abstieg**s ("Wasser fließt stets in Richtung $- \nabla F$ ")

Methode des steilsten Abstiegs (steepest descent)

Algorithmus:

1) Wähle ein $\mathbf{x}_1 = (x_{11}, \ldots, x_{1n})'$
2) Für k = 1, 2, 3, ...
    berechne die Richtung $\mathbf{d}_k := - (\nabla F(\mathbf{x}_k))'$ (muß jetzt noch nicht unbgedingt normiert sein.
    Minimiere $\varphi(\lambda) = F(\mathbf{x}_k + \lambda \mathbf{d}_k)$ durch "line search".
    Man erhält den optimalen Wert λ^*

und als nächsten Punkt $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda * \mathbf{d}_k$



Zunächst geht man wie das Wasser vor: man wählt die Richtung des steilsten Abstiegs. Diese Richtung behält man jedoch so lange bei, bis die Funktion F (diese entspricht der "Höhe") wieder anzusteigen beginnt. Dort sieht man sich erneut nach dem steilsten Abstieg um. (Gradient und Höhenschichtlinie sind dort immer orthogonal zueinander.
Warum kann man es nicht genauso wie das Wasser machen? Man müßte unendlich viele Neuberechnungen der Richtung machen (Schrittweite müßte infinitesimal klein werden) und würde dadurch unendliche Laufzeit erreichen. Unser Optimierungsverfahren ist hingegen schon recht effizient: Anfangs macht man große Schritte, erst wenn man sich dem Optimum nähert, werden die Schritte kleiner.

"Steepest descent" verläuft immer in aufeinanderfolgenden orthogonalen Richtungen. Dadurch ist man ziemlich eingeschränkt und das ist sicher noch nicht das optimale Verfahren.

Das mehrdimensionale Newtonverfahren

Hier wird das im vorhergehenden Kapitel schon angeführte Newton Verfahren für mehrere Dimensionen verallgemeinert.

Auch hier wollen wir F(**x**) wieder durch eine quadratische Funktion h(**x**) etwa im Punkt $\mathbf{x_1}$ annähern:

$$h(\mathbf{x}) = \mathbf{a} + \mathbf{b}'\,\mathbf{x} + \frac{1}{2}\mathbf{x}'\,\mathbf{G}\,\mathbf{x}$$

$$\nabla h(\mathbf{x}) = \mathbf{b}' + \mathbf{x}'\,\mathbf{G}$$

$$\mathbf{H}_h(\mathbf{x}) = \mathbf{G} \quad \text{(Hesse'sche Matrix von } h(\mathbf{x}))$$

Es muß also in $\mathbf{x_1}$ gelten:

$$F(\mathbf{x_1}) = h(\mathbf{x_1})$$

$$\nabla F(\mathbf{x_1}) = \nabla h(\mathbf{x_1})$$

$$\mathbf{H}_F(\mathbf{x_1}) = \mathbf{H}_h(\mathbf{x_1})$$

setzt man nun die Funktion h(**x**) wie oben definiert ein, so erhält man:

$$F(\mathbf{x_1}) = \mathbf{a} + \mathbf{b}'\,\mathbf{x_1} + \frac{1}{2}\mathbf{x_1}'\,\mathbf{G}\,\mathbf{x_1}$$

$$\nabla F(\mathbf{x_1}) = \mathbf{b}' + \mathbf{x_1}'\,\mathbf{G}$$

$$\mathbf{H}_F(\mathbf{x_1}) = \mathbf{G}$$

Die Matrix **G** ist damit gerade gleich der Hessematrix von F. Man kann nun G in der zweiten Zeile (erste Ableitung) einsetzen durch $\mathbf{H}_F(\mathbf{x}_1)$ und erhält, wenn man nach **b'** auflöst:

$\mathbf{b}' = \nabla F(\mathbf{x}_1) - \mathbf{x}_1' \mathbf{H}_F(\mathbf{x}_1)$ bzw.

$\mathbf{b} = (\nabla F(\mathbf{x}_1))' - \mathbf{H}_F(\mathbf{x}_1)\, \mathbf{x}_1$, da $\mathbf{H}_F(\mathbf{x}_1) = \mathbf{H}_F(\mathbf{x}_1)'$ (Hesse'sche Matrix ist symmetrisch).

Berechnen wir nun das Minimum **x*** der Funktion h(**x**):

$\nabla h(\mathbf{x}) = \mathbf{0}$

also $\mathbf{b}' + \mathbf{x}^{*\prime}\, \mathbf{G} = \mathbf{0}$

und daher $\mathbf{x}^* = -\mathbf{G}^{-1}\mathbf{b} = -(\mathbf{H}_F(\mathbf{x}_1))^{-1} ((\nabla F(\mathbf{x}_1))' - \mathbf{H}_F(\mathbf{x}_1)\, \mathbf{x}_1) = \mathbf{x}_1 - (\mathbf{H}_F(\mathbf{x}_1))^{-1} (\nabla F(\mathbf{x}_1))'$

Daraus ergibt sich der Algorithmus von Newton und Raphson für den mehrdimensionalen Fall:

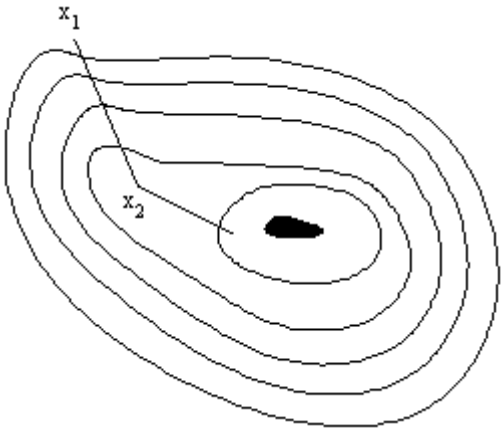
Algorithmus:

1) Wähle ein $\mathbf{x}_1 = (x_{11}, \ldots , x_{1n})'$

2a) Für k = 1, 2, 3, ...

$\mathbf{x}_{k+1} = \mathbf{x}_k - (\mathbf{H}_F(\mathbf{x}_k))^{-1} (\nabla F(\mathbf{x}_k))'$

2b) Wiederhole 2a) bist die gewünschte Genauigkeit erreicht ist.



Vorteil: Man geht gleich in eine "gute Richtung" (bei quadratischer ZF genau in Richtung des Minimums), statt einen Zick-Zack-Kurs zu steuern.

Nachteile:

- Die 2. Ableitungen müssen bekannt sein.
- Die Hesse'sche Matrix muß immer positiv-definit sein (entspricht strikter Konvexität)
- Der Funktionswert kann zwischendurch sogar ansteigen (das ist bei "steepest descent" nie der Fall), dies kann man durch Kombination mit "line search" vermeiden.
- Die Inversion der Hesse'schen Matrix kann numerisch aufwändig sein.

Das Newton-Raphson Verfahren kann also nur dann angewendet werden, wenn zweite Ableitungen bekannt / bestimmbar sind.

Beispiel:

$F(x,y) = x^2 + 4y^2 - 4x - 8y \rightarrow \min!$

$\nabla F = (2x - 4, 8y - 8)$

$\nabla F = (2x - 4, 8y - 8)$

$\mathbf{H}_F = \begin{pmatrix} 2 & 0 \\ 0 & 8 \end{pmatrix}$ und $\mathbf{H}_F^{-1} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{8} \end{pmatrix}$

Newton-Raphson Algorithmus:

Startvektor (0, 0)'

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

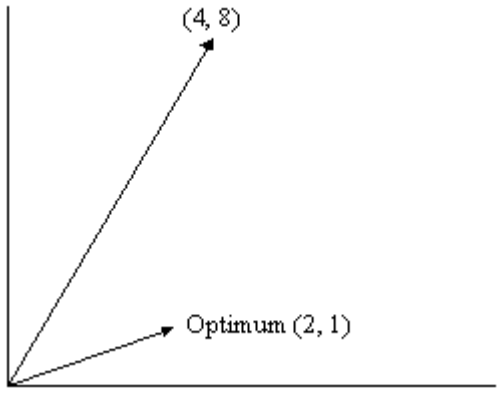
$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{8} \end{pmatrix} \begin{pmatrix} -4 \\ -8 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} - \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{8} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Der Funktionswert verändert sich nicht mehr, man ist bereits im Optimum.

Steepest Descent Algorithmus:

Startvektor (0, 0)'
$\mathbf{d}_1 := - (\nabla F(\mathbf{x}_1))' = - (-4 -8)' = (4\ 8)'$
Das heißt, man geht zunächst in eine "falsche" Richtung. Erst nach einer Reihe von Rechenschritten, kommt man wieder in die richtige Gegend.



5. Nichtlineare Optimierung mit Nebenbedingungen

5.1. Einleitung

Zielfunktion F: $\mathbf{R}^n \rightarrow \mathbf{R}$

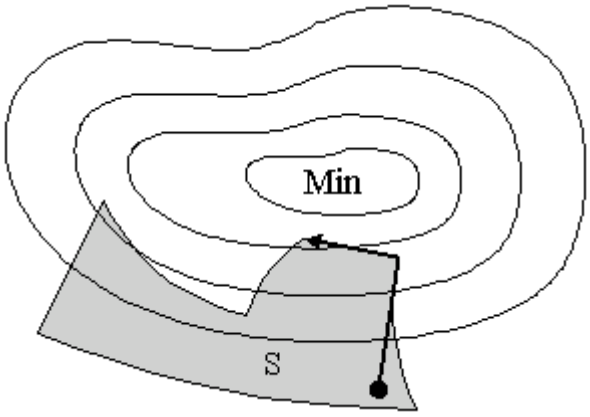
$F(x) \rightarrow \min!$
$x \in \mathbf{S}$

wobei $\mathbf{S} \subset \mathbf{R}^n$

Meist ist die zulässige Menge **S** dadurch gegeben, daß gewisse Ungleichungen und/oder Gleichungen erfüllt sein sollen.
Wir beginnen nun zunächst mit der Situation, wo nur Ungleichungen vorliegen, also:

$F(x) \rightarrow \min!$
$g_1 \leq 0$
.
.
$g_m \leq 0$

(bringen also in den Nebenbedingungen alles auf die linke Seite, eventuell mit -1 multiplizieren)

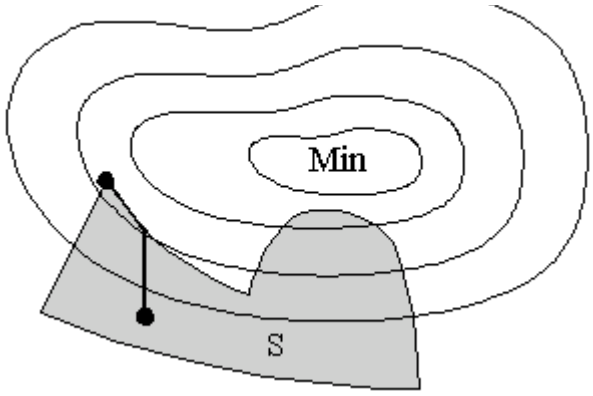


Anschaulich etwa im Fall n=2: Man sucht den tiefsten Punkt in einem Gelände, ist aber nun zusätzlich durch "Zäune" eingeschränkt. Bei $g_i = 0$ verläuft der "Zaun". Auf einer Seite des "Zaunes" ($g_i \leq 0$) ist man in der zulässigen Menge, auf der anderen Seite ($g_1 > 0$) in der unzulässigen.

Man könnte nun versuchen ein Steepest-descent-Verfahren an diese Situation zu adaptieren:

- Gehe in Richtung des negativen Gradienten
- Wenn man an einen "Zaun" anstößt, gehe in Richtung fallender Funktionswerte den "Zaun" entlang, solange bis der negative Gradient wieder ins Innere weist.

Aber: Sogar im Fall einer unimodalen Zielfunktion führt diese Verfahren nicht immer zum globalen Optimum. Das Ergebnis ist abhängig vom gewählten Startwert.

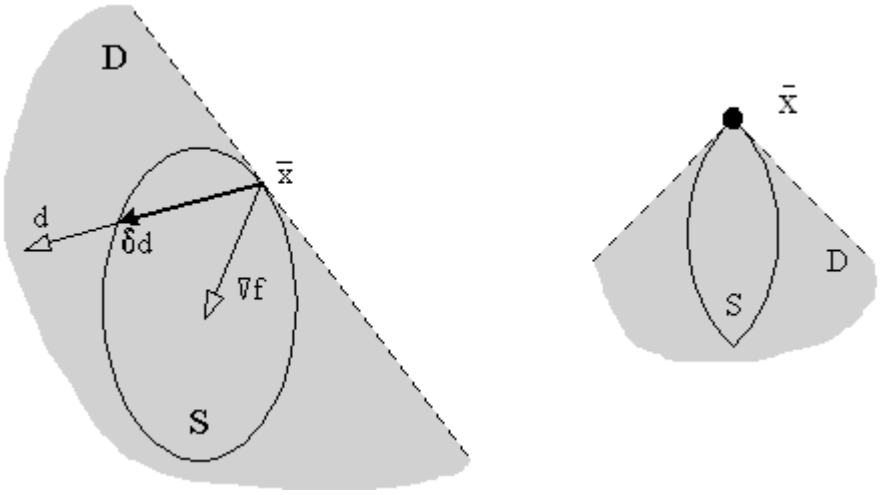


Obwohl man also das globale Optimum i.a. nicht auf Anhieb finden kann, erscheint es sinnvoll, solche Punkte zu erkennen, die als Kandidaten für das Optimum in Frage kommen.
Dies leisten sogenannte **notwendige Optimalitätsbedingungen**. In gewissen Sondersituationen ist ein Punkt, der eine Optimalitätsbedingung erfüllt, auch mit Sicherheit bereits optimal. In diesem Fall spricht man von einer hinreichenden Optimalitätsbedingung.

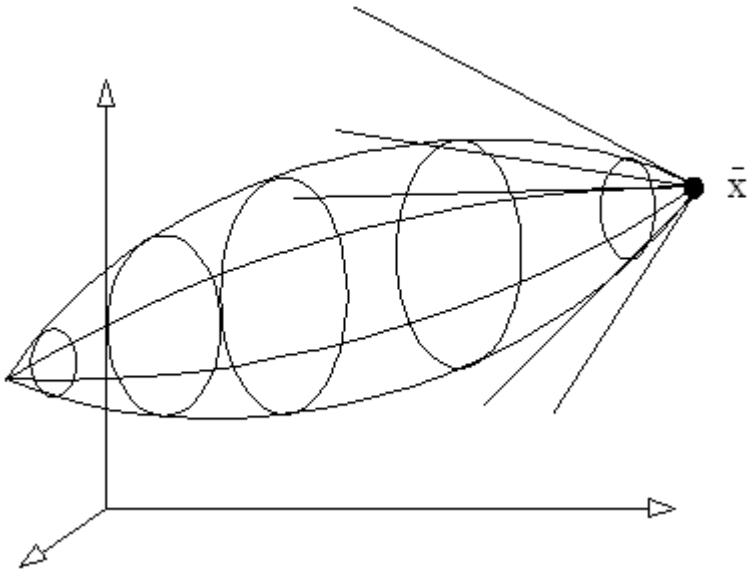
5.2. Optimalitätsbedingungen

Zunächst: Geometrische Optimalitätsbedingungen

Definition:
Sei $\mathbf{S} \subseteq \mathbf{R}^n$ und $\overline{x} \in \mathbf{S}$. Der **Kegel zulässiger Richtungen** von **S** an $\overline{x}$ ist gegeben durch die Menge **D** aller Vektoren $\mathbf{d} \neq \mathbf{0}$ für die gilt:
Es gibt ein $\delta > 0$, sodaß $\overline{x} + \lambda\,\mathbf{d} \in \mathbf{S}$ für alle λ mit $0 < \lambda < \delta$.
(d.h. eine hinreichend kurze Strecke in Richtung **d** ist zur Gänze in **S** enthalten)



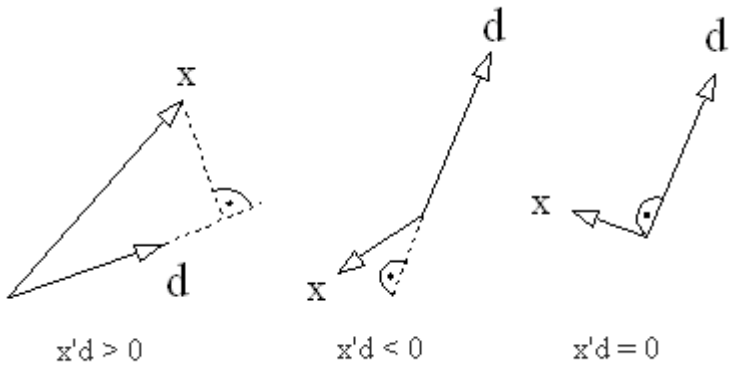
Im $\mathbf{R}^3$ etwa (hier ist D tatsächlich ein Kegel im üblichen Sinn):



Die Richtungen $\mathbf{d} \in \mathbf{D}$ heißen **zulässige Richtungen**. Eine **kleine** Bewegung von $\overline{x}$ aus in irgendeine zulässige Richtung führt wieder auf einen zulässigen Punkt.

Verwenden nun Methoden der linearen Algebra, um festzustellen ob ein beliebiger Richtungsvektor **d** an einer Stelle **x** in Richtung eine Anstieges oder Abstieges zeigt.
Sei **d** ein Einheitsvektor (d.h. $\|\mathbf{d}\| = 1$), so ist $\mathbf{x}'\mathbf{d}$ die Länge der Projektion des Vektors **x** auf den Vektor (bzw. die Richtung) **d**:

Vektor (bzw. die Richtung) **d**.

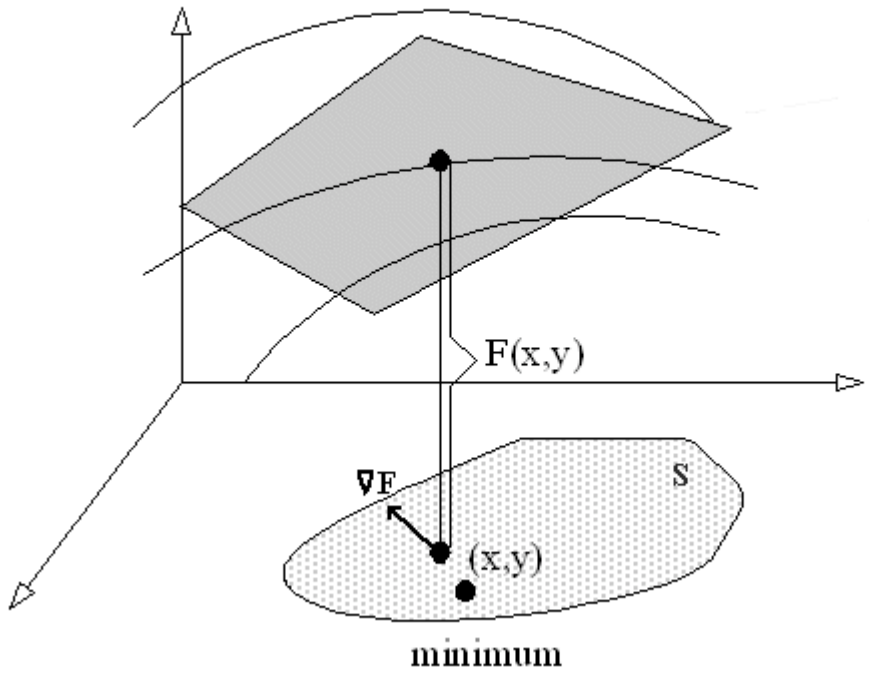


Wir hatten festgestellt: ∇F bzw -∇F zeigen in Richtung des steilsten Anstiegs bzw. des steilsten Abstiegs.

Darüberhinaus gilt:
Falls $\nabla F(\overline{x})$**d** > 0 bzw. < 0, dann ist **d** an der Stelle $\overline{x}$ eine Anstiegs- bzw. Abstiegsrichtung.

Oder genauer:
Falls $\nabla F(\overline{x})$**d** > 0 , gibt es ein δ > 0, sodaß F($\overline{x}$ + λ**d) > F(** $\overline{x}$ **)** für alle λ mit 0 < λ < δ.
(Analog für Abstiegsrichtungen)

Geometrisch gesehen approximieren wir F durch eine Tangentialebene im Punkt $\overline{x}$ (lineare Approximation). In der näheren Umgebung von $\overline{x}$ verhält sich diese Approximation so ähnlich wie die Funktion F(x) selbst.



Also: Richtungen **d** mit $\nabla F(\overline{x})\mathbf{d} < 0$ verbessern den Funktionswert, solche mit $\nabla F(\overline{x})\mathbf{d} > 0$ verschlechtern ihn!

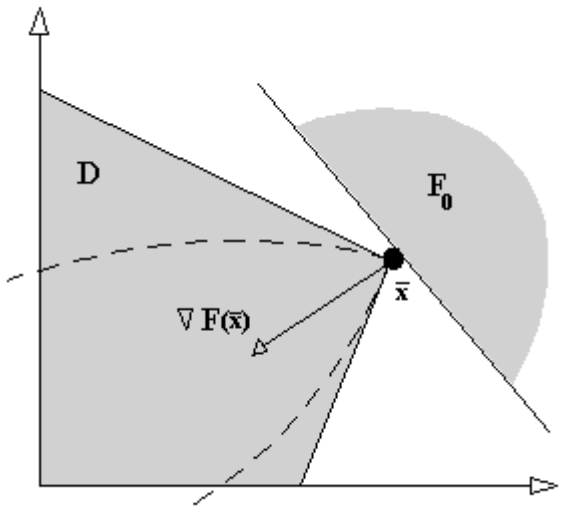
Bezeichne mit $F_0 := \{\mathbf{d} \mid \nabla F(\overline{x})\mathbf{d} < 0\}$ die Menge aller "verbessernden" Richtungen.

Theorem:
Wenn $\overline{x}$ ein lokales Optimum ist, so gilt: $F_0 \cap D = \{\}$

Beweis:
Angenommen **x** sei ein lokales Optimum, aber der Durchschnitt nicht leer, d.h. es existiert ein Vektor $\mathbf{d} \in F_0 \cap D$

Wegen $\mathbf{d} \in F_0$ gilt: $\nabla F(\overline{x})\mathbf{d} < 0$, also nach obigem $F(\overline{x} + \lambda \mathbf{d}) < F(\overline{x})$ für hinreichend kleines λ.
Wegen $\mathbf{d} \in D$ gilt: $\overline{x} + \lambda \mathbf{d} \in \mathbf{S}$ für hinreichend kleines λ (nach der Definition von D)

Damit gibt es aber in einer beliebig kleinen Umgebung von $\overline{x}$ zulässige Punkte mit besserem Funktionswert, das heißt aber $\overline{x}$ kann kein lokales Optimum gewesen sein. Widerspruch!



F_0 und D haben keinen Punkt gemeinsam ($\overline{x}$ selbst entspricht dem Nullvektor und dieser gehört nicht zu F_0)

Die Zugehörigkeit eines Vektors **d** zur Menge F_0 kann rechnerisch leicht geprüft werden, nicht aber die zur Menge D.
Daher suchen wir eine Menge $G_0 \subseteq D$ für die sich die Zugehörigkeit leichter prüfen läßt.

$F_0 \cap G_0 = \{\}$ wäre dann wieder eine notwendige Optimalitätsbedingung, denn falls $\overline{x}$ ein lokales Optimum ist, so ist ja $F_0 \cap D = \{\}$ und damit auch $F_0 \cap G_0 = \{\}$, denn es ist ja $G_0 \subseteq D$.

Sei $\overline{x} \in \mathbf{S}$. In $\overline{x}$ sind gewisse Nebenbedingungen $g_i(\overline{x}) \leq 0$ **aktiv**, nämlich die mit $g_i(\overline{x}) = 0$. Andere sind **passiv**, nämlich die mit $g_i(\overline{x}) < 0$.
Sei $I = I(\overline{x}) = \{i \mid g_i(\overline{x}) = 0\}$ die Indexmenge aller aktiven Nebenbedingungen an der Stelle

Sei $I = I(\overline{x}) = \{i \mid g_i(\overline{x}) = 0\}$ die Indexmenge aller aktiven Nebenbedingungen an der Stelle $\overline{x}$.

Im Folgenden sollen F und g_i (i =1,...,m) in $\overline{x}$ differenzierbar sein.

Theorem:

Wenn $\overline{x}$ ein lokales Optimum ist, so gilt $F_0 \cap G_0 = \{\}$, wobei

$F_0 = \{\mathbf{d} \mid \nabla F(\overline{x})\mathbf{d} < 0\}$ und

$G_0 = \{\mathbf{d} \mid \nabla g_i(\overline{x})\mathbf{d} < 0$ für alle $i \in I\}$

Beweis:

Es existiere ein Vektor $\mathbf{d} \in F_0 \cap G_0$. Aus $\nabla F(\overline{x})\mathbf{d} < 0$ folgt dann $F(\overline{x} + \lambda \mathbf{d}) < F(\overline{x})$ für hinreichend kleines λ, d.h. es existieren Werte $\overline{x} + \lambda \mathbf{d}$ mit besser Zielfunktion in jeder Umgebung von $\overline{x}$.

Zu zeigen bleibt deren Zulässigkeit.

1) Sei $i \in I$, d.h. $g_i(\overline{x}) = 0$

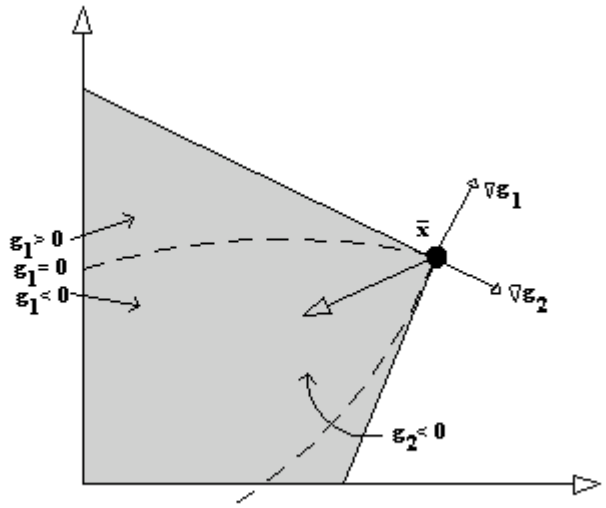
   Dann gilt wegen $\nabla g_i(\overline{x})\mathbf{d} < 0$: $g_i(\overline{x} + \lambda \mathbf{d}) < g_i(\overline{x})$ für hinreichend kleines λ.

   $g_i(\overline{x})$ ist aber gleich 0, daher ist $g_i(\overline{x} + \lambda \mathbf{d}) < 0$.

2) Sei $i \notin I$, d.h. $g_i(\overline{x}) < 0$

   Es ist g_i differenzierbar in $\overline{x}$, also auch stetig in $\overline{x}$ und daher ist auch $g_i(\overline{x} + \lambda \mathbf{d}) < 0$ für hinreichend kleines λ.

In einer beliebig kleinen Umgebung von $\overline{x}$ gibt es also zulässige Punkte mit besserem Zielfunktions-Wert. Das ist im Widerspruch zur Annahme!



Hier ist sogar $G_0 = D$, im allgemeinen ist aber $G_0 \subseteq D$.

Beispiel 5.1:

$F(x_1,x_2) = (x_1 - 3)^2 + (x_2 - 2)^2 \rightarrow$ min !

$g_1(x_1,x_2) = x_1^2 + x_2^2 - 5$

$g_2(x_1,x_2) = x_1 + x_2 - 3$

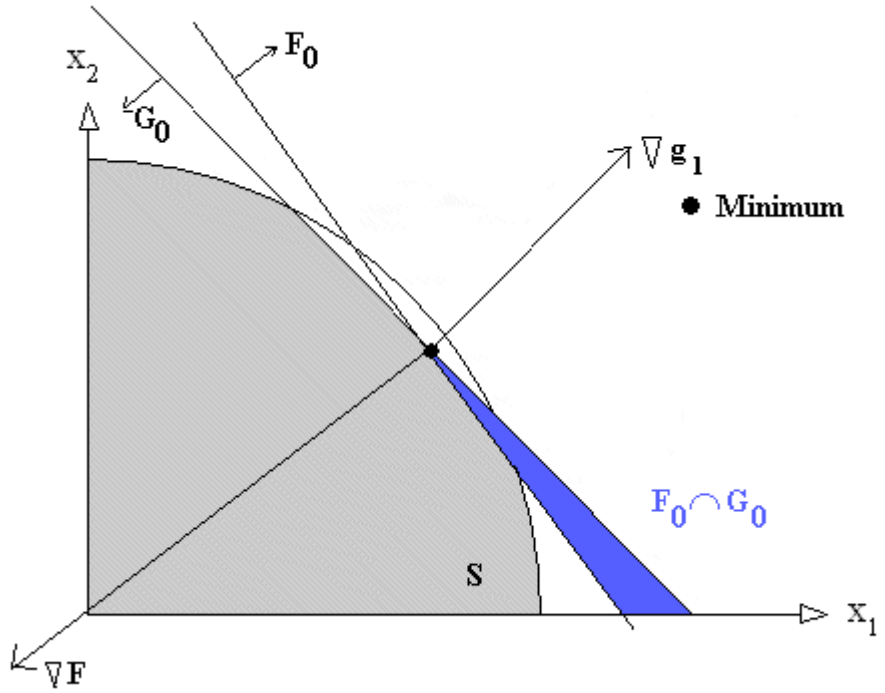
$g_2(x_1,x_2) = 2 - x_1 - x_2$
$g_3(x_1,x_2) = -x_1$
$g_4(x_1,x_2) = -x_2$

(die Vorzeichenbeschränkungen müssen jetzt explizit ausgedrückt werden)

Wir betrachten nun folgende zwei Punkte (9/5, 6/5) und (2, 1).
Im ersten Punkt ist nur g_2 aktiv, also I = {2}.
Im zweiten Punkt sind g_1 und g_2 aktiv, also I = {1,2}.

Berechne die Gradienten:
$\nabla F(x_1,x) = (2(x_1 - 3), 2 (x_2 - 2))$
$\nabla g_1(x_1,x_2) = (2x_1, 2x_2)$
$\nabla g_2(x_1,x_2) = (1, 1)$
(die anderen Nebenbedingungen sind hier nicht aktiv)

Also gilt im Punkt (9/5, 6/5):
$\nabla F(9/5, 6/5) = (-12/5, -8/5)$
$\nabla g_2(9/5, 6/5) = (1, 1)$
$F_0 = \{\mathbf{d} \mid -12/5\ d_1 - 8/5\ d_2 < 0 \}$ und
$G_0 = \{\mathbf{d} \mid\ d_1 + d_2 < 0 \}$
Das sind zwei Halbebenen und ihr Durchschnitt ist nicht leer! Daher ist (9/5, 6/5) sicher kein Optimum.



Im Punkt (2, 1):
$\nabla F(2, 1) = (-2, -2)$

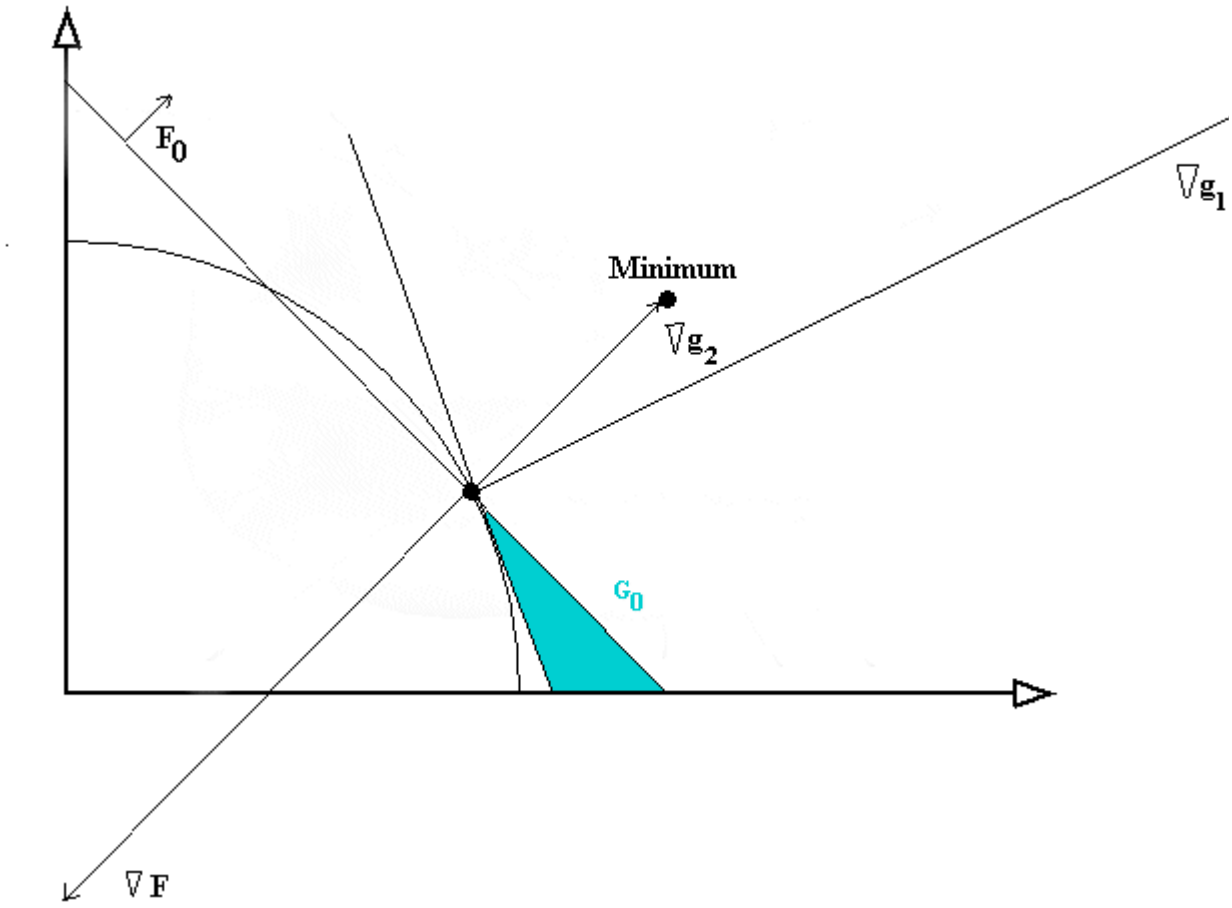
$\nabla g_1(2, 1) = (4, 2)$

$\nabla g_2(2, 1) = (1, 1)$

$F_0 = \{\mathbf{d} \mid -2d_1 - 2d_2 < 0\}$ und

$G_0 = \{\mathbf{d} \mid 4d_1 + 2d_2 < 0$ und $d_1 + d_2 < 0\}$

Der Durchschnitt dieser Mengen ist leer! Daher kann (2, 1) ein Optimum sein (muß es aber nicht).



Die notwendige Optimalitätsbedingung $F_0 \cap G_0 = \{\}$ erlaubt es bereits, rechnerisch zu prüfen, ob ein $\overline{x} \in \mathbf{S}$ als Kandidat für die optimale Lösung überhaupt in Frage kommt. Sie erlaubt aber nicht , solche $\overline{x}$ zu finden. Daher formt man diese Optimalitätsbedingungen weiter um. Das führt uns über die **Fritz-John-Bedingungen** schließlich zu den **Kuhn-Tucker-Bedingungen**.

5.3. Fritz-John Bedingungen

Wir brauchen zunächst einen Hilfssatz aus der Theorie konvexer Mengen:

Theorem (Satz von Gordan):
Sei A eine (nxm) Matrix. Genau eines der folgenden beiden Systeme besitzt eine Lösung:
System 1: **Ax < 0**
System 2: **A'u = 0, u $\geq$ 0, u $\neq$ 0**
(hier ohne Beweis)

Beispiel:

$A = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ (da A eine (2x1) Matrix ist, muss x hier ein (1x1) Vektor sein!)

System 1: Es müßte gleichzeitig gelten x < 0 und -x < 0, daher keine Lösung

System 2: $\begin{pmatrix} -1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = 0$, **u $\geq$ 0, u $\neq$ 0.** Man erhält als Lösung $u_1 = 1$ und $u_2 = 1$.

Im folgenden setzen wir voraus, daß F und g_i (i=1,..m) in $\overline{x}$ differenzierbar sind.
I sei die Menge der Indizes der aktiven Nebenbedingungen

Theorem (Fritz-John Optimalitätsbedingungen):
Wenn $\overline{x}$ lokales Optimum ist, so gibt es Skalare u_0 und u_i (i $\in$ I), sodaß
(1) $u_0 (\nabla F(\overline{x}))' + \sum_{i \in I} u_i (\nabla g_i(\overline{x}))' = 0$
(2) u_0, $u_i \geq 0$ (i $\in$ I)
(3) nicht alle Skalare u_0, u_i (i $\in$ I) sind gleich 0.

Äquivalent dazu wären folgende Bedingungen:
(a) $u_0 (\nabla F(\overline{x}))' + \sum_{i \in I} u_i (\nabla g_i(\overline{x}))' = 0$
(b) $u_i \cdot g_i(\overline{x})$ (i = 1, ..., m)
(c) u_0, $u_i \geq 0$ (i = 1, ..., m)
(d) nicht alle Skalare u_0, u_i (i = 1, ..., m) sind gleich 0.
(der Vorteil dieser Schreibweise ist, daß man nicht zwischen aktiven und passiven Nebenbedingungen unterscheiden muß)

Beweis:
Haben bereits gesehen: $\overline{x}$ lokales Optimum $\Longrightarrow$ $F_0 \cap G_0 = \{\}$.
D.h.: es gibt keinen Vektor **d**, sodaß
d $\in F_0$, also $\nabla F(\overline{x})\mathbf{d} < 0$ **und**
d $\in G_0$, also $\nabla g_i(\overline{x})\mathbf{d} < 0$ für alle i $\in$ I.

OBdA sei nun I = {1,...., k} (k $\leq$ m), d.h. g_1 ,...., g_k sind aktiv und g_{k+1} ,...., g_m sind passiv.

$\begin{pmatrix} \nabla F(\overline{x}) \end{pmatrix}$

Sei $A := \begin{pmatrix} \nabla F(\overline{x}) \\ \nabla g_1(\overline{x}) \\ . \\ . \\ \nabla g_k(\overline{x}) \end{pmatrix}$

Dann gilt nach obigem:
Es gibt kein **d**, sodaß **Ad < 0**. D.h. **Ad < 0** ist nicht lösbar.
Nach dem Satz von Gordan muß dann aber das System 2 (**A'u = 0, u** $\geq$ **0, u** $\neq$ **0**) eine Lösung besitzen.

$$\left((\nabla F(\overline{x}))', \quad (\nabla g_1(\overline{x}))', \quad \dots \quad , (\nabla g_k(\overline{x}))' \right) \begin{pmatrix} u_0 \\ u_1 \\ . \\ . \\ u_k \end{pmatrix} = 0$$

also $(\nabla F(\overline{x}))' u_0 + (\nabla g_1(\overline{x}))' u_1 + \dots + (\nabla g_k(\overline{x}))' u_k = 0,$
wobei u_0 , u_1 , ..., $u_k \geq 0$
und
$(u_0$, u_1 , ..., u_k) $\neq$ (0, 0, ..., 0).
Das sind aber gerade die Bedingungen (1) bis (3).

Zeigen nun noch, daß aus (1) - (3) folgt (a) - (d):
Sei wieder oBdA I = {1,...., k}
Seien u_0 , u_1 , ..., u_k die Lösungen von (1) - (3). Wir setzen u_{k+1} =....= u_m = 0.
Dann gilt offenbar (a), da man der Summe in (1) ja nur verschwindende Terme hinzufügt.
Außerdem gilt (b), denn für i $\in$ I gilt g_i ($\overline{x}$) = 0 und für i $\notin$ I ist u_i = 0.
(c) und (d) sind ebenfalls erfüllt.

Analog kann man sich auch überlegen: aus (a) - (d) folgt (1) - (3)

Damit hat man gezeigt, daß (1) - (3) und (a) - (d) äquivalent sind.

Die Skalare u_0 , u_i heißen **Lagrange'sche Multiplikatoren**. (Kommen unter diesem Namen ursprünglich beim Lösen von Problemen mit Gleichheits-Nebenbedingungen vor)

Beispiel 5.2:

$F(x_1,x_2) = (x_1 - 3)^2 + (x_2 - 2)^2 \rightarrow$ min !
$g_1(x_1,x_2) = x_1^2 + x_2^2 - 5 \leq 0$
$g_2(x_1,x_2) = x_1 + 2x_2 - 4 \leq 0$
$g_3(x_1,x_2) = -x_1 \leq 0$
$g_4(x_1,x_2) = -x_2 \leq 0$

$\nabla F(x_1,x) = (2(x_1 - 3), 2 (x_2 - 2))$
$\nabla g_1(x_1,x_2) = (2x_1, 2x_2)$
$\nabla g_2(x_1,x_2) = (1, 2)$
$\nabla g_3(x_1,x_2) = (-1, 0)$
$\nabla g_4(x_1,x_2) = (0, -1)$

Frage 1: Kann (0, 0)' ein optimaler Punkt sein ?
Aktiv sind g_3 und g_4, also I = {3, 4}.

Fritz-John-Bedingungen:

(1) $u_0 (-6, -4)' + u_3 (-1, 0)' + u_4 (0, -1)' = (0, 0)'$
(2) $u_0 , u_3 , u_4 \geq 0$
(3) $(u_0 , u_3 , u_4) \neq (0, 0, 0)$.

Man erhält also
$u_3 = -6 u_0$
$u_4 = -4 u_0$
Da $u_0 \geq 0$ sein muß, kann man 2 Fälle unterscheiden:

Falls $u_0 = 0$: dann ist $u_0 = u_3 = u_4 = 0$, das darf aber laut (3) nicht sein.
Falls $u_0 > 0$: dann ist $u_3 < 0$ und $u_4 < 0$, das darf aber laut (2) nicht der Fall sein.

Daraus folgt: (0, 0)' ist **kein** optimaler Punkt

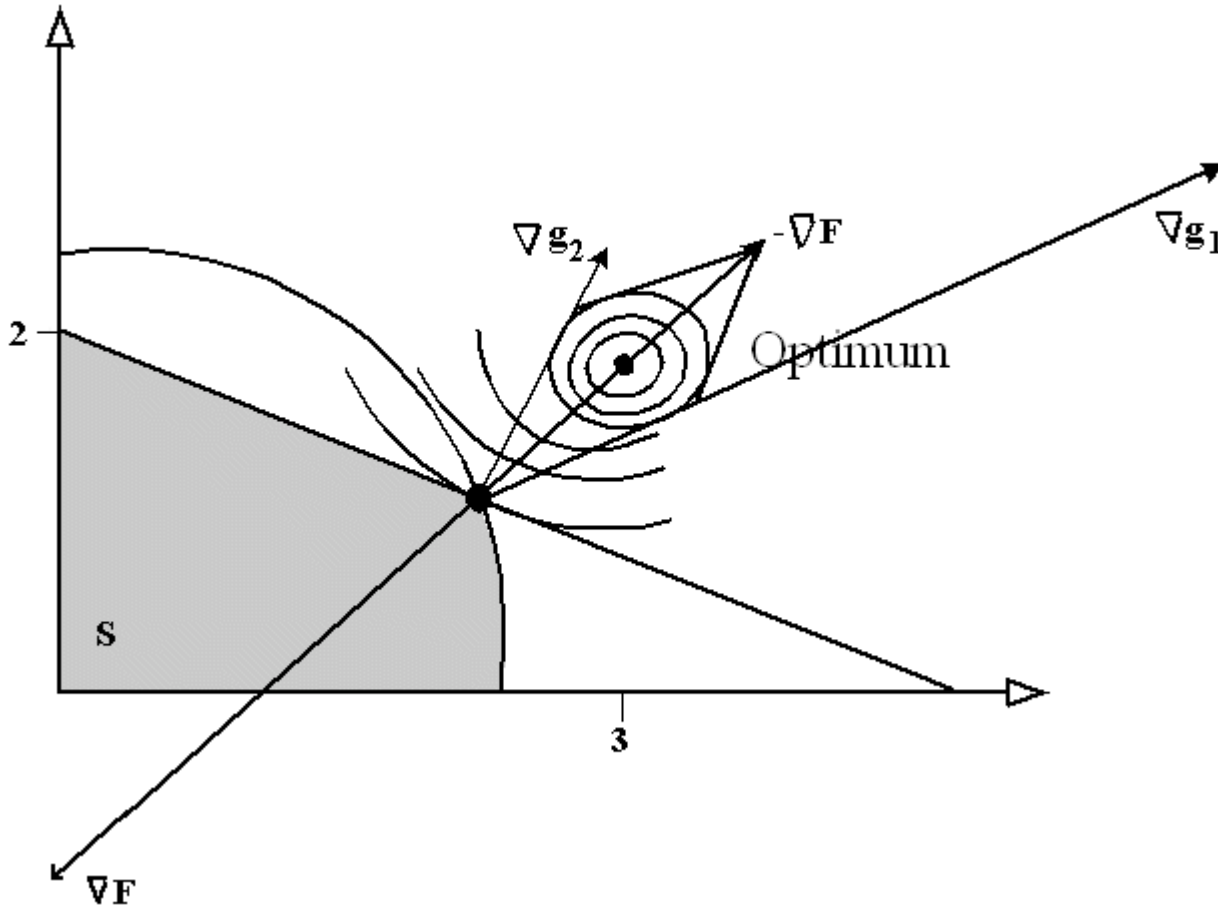
Frage 2: Kann (2, 1)' ein optimaler Punkt sein ?
Aktiv sind g_1 und g_2, also I = {1, 2}.

Fritz-John-Bedingungen:

(1) $u_0 (-2, -2)' + u_1 (4, 2)' + u_2 (1, 2)' = (0, 0)'$
(2) $u_0 , u_1 , u_2 \geq 0$
(3) $(u_0 , u_1 , u_2) \neq (0, 0, 0)$.

Aus dem Gleichungssystem

$4u_1 + u_2 = 2 u_0$
$2u_1 + 2u_2 = 2 u_0$
erhalten wir $u_1 = 1/3 u_0$ und $u_2 = 2/3 u_0$
Setzt man nun etwa $u_0 = 1$, dann ist $u_1 = 1/3$ und $u_2 = 2/3$.
Die Fritz-John Bedingungen sind erfüllt und der Punkt (2, 1)' ist möglicherweise ein optimaler Punkt (in diesem Fall ist er es auch tatsächlich).



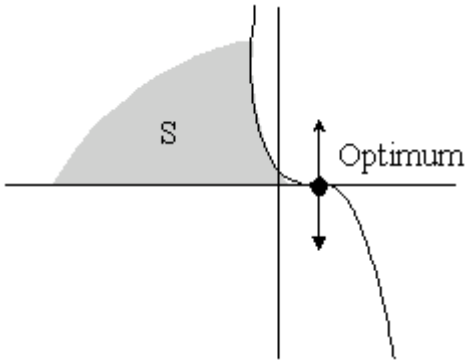
Es gilt also $-\nabla F(\overline{x}) = \frac{1}{3}\nabla g_1(\overline{x}) + \frac{2}{3}\nabla g_2(\overline{x})$ (d.h. der negative Gradient ist darstellbar als positive Linearkombination der Nebenbedingungs-Gradienten, er liegt "zwischen" diesen)

Beispiel 5.3:

$F(x_1,x_2) = -x_1 \rightarrow$ min !

$g_1(x_1,x_2) = x_2 - (1 - x_1)^3 \leq 0$

$g_2(x_1,x_2) = -x_2 \leq 0$



$\nabla F(x_1,x) = (-1, 0)$

$\nabla g_1(x_1,x_2) = (3(1 - x_1)^2, 1)$

$\nabla g_2(x_1,x_2) = (0, -1)$

Offensichtlich ist (1, 0)' das Optimum. Kontrolle mittels Fritz-John:
Frage: Kann (1, 0)' ein optimaler Punkt sein ?
Aktiv sind g_1 und g_2, also I = {1, 2}.

Fritz-John-Bedingungen:

(1) u_0 (-1, 0)' + u_1 (0, 1)' + u_2 (0, -1)' = (0, 0)'
(2) u_0 , u_1 , $u_2 \geq 0$
(3) (u_0 , u_1 , u_2) $\neq$ (0, 0, 0).

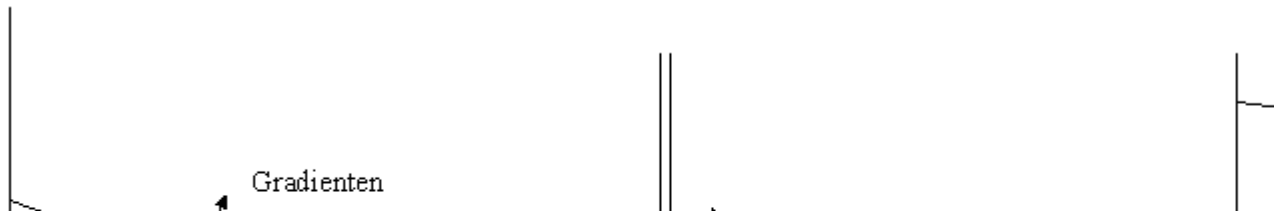
Man erhält $u_0 = 0$, u_1 und u_2 können beliebige Werte annehmen, also etwa $u_1 = u_2 = 1$
Die notwendigen Optimalitätsbedingungen sind damit erfüllt

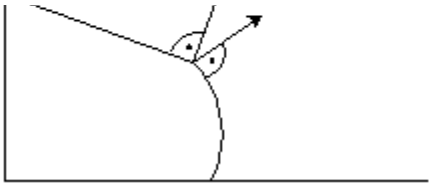
Auf den ersten Blick ist also alles in Ordnung. Hätten wir als Zielfunktion allerdings $F(x_1,x_2) = x_1$ als Zielfunktion gewählt, hätten wir das selbe Ergebnis erhalten! Und das ist sicher kein Optimum.
Insbesondere hätte jede beliebige Zielfunktion die Fritz-John Bedingungen erfüllt, da $u_0 = 0$ und damit jede Information über die Zielfunktion "ausgeblendet" wurde.
Es bleibt als Bedingung nur mehr: $\sum_{i \in I} (\nabla g_i(\overline{x}))' = 0$
Das sagt also nur mehr aus, daß die Nebenbedingungen im gewissen Sinne linear abhängig sind (es gibt eine nichtnegative, nichttriviale Linearkombination, die den Nullvektor ergibt).

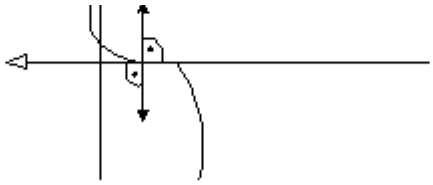
Beispiel 5.4:
Sei g_i an der Stelle $\overline{x}$ aktiv und der Gradient $\nabla g_i(\overline{x}) = \mathbf{0}$
Setze $u_i = 1$ und alle anderen $u_j = 0$, dann ist die Fritz-John Bedingung auf jeden Fall erfüllt.

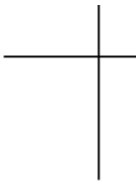
Das bedeutet also, daß die Fritz-John Bedingungen zu viele Kandidaten für das Optimum zulassen!

Um das zu vermeiden, wollen wir erzwingen, daß $u_0 > 0$ sein muß.
Das geht, indem man den Nebenbedingungen sogenannte "**constraint qualifications**" auferlegt.
Es gibt mehrere mögliche constraint qualifications, die $u_0 > 0$ nach sich ziehen. Eine Möglichkeit liegt nach den obigen Überlegungen nahe:
Man verlangt, daß im Optimum die Gradienten $\nabla g_i(\overline{x})$ der aktiven Nebenbedingungen **linear unabhängig** sind.
Damit scheiden die Beispiele 5.3 und 5.4 aus, das Beispiel 5.2 kann aber weiterhin behandelt werden.









5.4. Kuhn-Tucker Bedingungen

Die Kuhn-Tucker Bedingungen nennt man auch Karush-Kuhn-Tucker Bedingungen (KKT). Unter den KKT Bedingungen versteht man Fritz-John Bedingungen + constraint qualifications.

Theorem (Kuhn-Tucker Optimalitätsbedingungen):
Es seien $\nabla g_i(\overline{x})$ (i ∈ I) linear unabhängig. Wenn $\overline{x}$ lokales Optimum ist, so gibt es Skalare u_i (i ∈ I), sodaß
(1) $(\nabla F(\overline{x}))' + \sum_{i \in I} u_i (\nabla g_i(\overline{x}))' = 0$
(2) $u_i \geq 0$ (i ∈ I)

Äquivalent dazu wären folgende Bedingungen:
(a) $(\nabla F(\overline{x}))' + \sum_{i \in I} u_i (\nabla g_i(\overline{x}))' = 0$
(b) $u_i \cdot g_i(\overline{x}) = 0$ (i = 1, ..., m)
(c) $u_i \geq 0$ (i = 1, ..., m)

Beweis:
Nach den Fritz-John Bedingungen müssen für ein optimales $\overline{x}$ Skalare v_0 , v_i (i ∈ I) existieren, sodaß

$$v_0(\nabla F(\overline{x}))' + \sum_{i \in I} v_i(\nabla g_i(\overline{x}))' = 0,$$

mit v_0 , $v_i \geq 0$ (i ∈ I) und nicht alle gleich 0.
Weiters muß $v_0 > 0$ gelten, denn im Fall $v_0 = 0$ wäre $\sum_{i \in I} v_i(\nabla g_i(\overline{x}))' = 0$, wobei nicht alle $v_i = 0$, sind, d.h. die Gradienten der g_i wären linear unabhängig.
Also kann man die Gleichung durch v_0 dividieren. Mit $u_i = v_i / v_0$ erhält man (1) und klarerweise gilt auch (2).
((a)-(c) anstelle von (1)-(2) erhält man wiederum, indem man $u_i := 0$ für $i \notin I$ setzt).

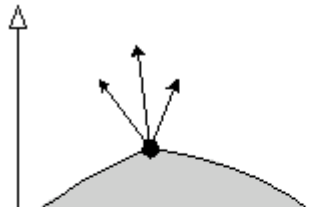
Die u_i heißen - so wie früher - **Lagrange'sche Multiplikatoren**.

Geometrische Interpretation der Kuhn-Tucker Bedingungen:

$$-\nabla F = \sum_{i \in I} u_i \nabla g_i,$$

d.h. der negative Gradientenvektor läßt sich als "nichtnegative Linearkombination" der Gradienten der aktiven Nebenbedingungen darstellen.

Das bedeutet geometrisch: $-\nabla F(\overline{x})$ liegt im "Kegel" der von $\nabla g_i(\overline{x})$ (i ∈ I) aufgespannt wird.



Der Punkt x_1 erfüllt die KKT Bedingungen, der Punkt x_2 erfüllt sie nicht.

Beispiel 5.5:
Eine Einheit soll auf 2 Anteile aufgeteilt werden. Anteil 1 wird maximiert, Anteil 2 darf aber nicht negativ werden. Sehr einfaches Beispiel.

$F(x_1,x_2) = - x_1 \rightarrow$ min !

$g_1(x_1,x_2) = x_1 + x_2 - 1 \leq 0$

$g_2(x_1,x_2) = \quad - x_2 \leq 0$

$\nabla F(x_1,x) = (-1, 0)$

$\nabla g_1(x_1,x_2) = (1, 1)$

$\nabla g_2(x_1,x_2) = (0, -1)$

Kuhn-Tucker Bedingungen (Version (a)-(c)):

$(-1, 0)' + u_1 (1, 1)' + u_2 (0, -1)' = (0, 0)'$

$u_1 (x_1 + x_2 - 1) = 0$

$u_2 (- x_2) = 0$

$u_1 , \ u_2 \geq 0$

Man erhält also 4 Gleichungen mit 4 Variablen. Gleichung 3 und 4 sind aber nichtlinear!
Fallunterscheidung:

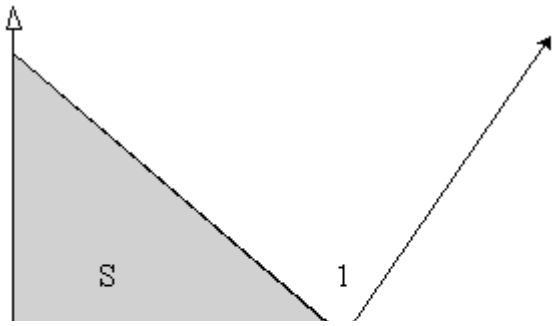
1) $u_1 = 0$: daraus würde wegen der ersten Gleichung folgen, daß -1 = 0, geht also nicht.

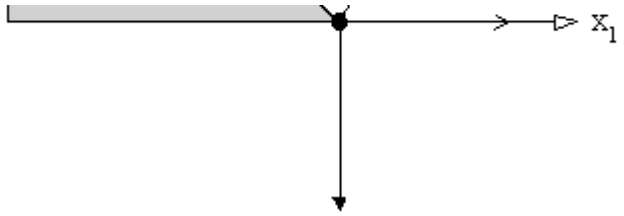
2) $u_1 \neq 0$: daraus folgt, daß $x_1 + x_2 - 1 = 0$. Weiter Unterscheidung

2a) $u_2 = 0$: daraus würde wegen der zweiten Gleichung folgen, daß $u_1 = 0$, geht also nicht, da ja $u_1 \neq 0$ laut Voraussetzung.

2b) $u_2 \neq 0$: daraus folgt, daß $x_2 = 0$ und in der Folge, daß $x_1 = 1$.

Also ist (1, 0)' der einzige Kandidat für ein Optimum und daher bereits die "optimale Lösung" (die Gradienten der aktiven Nebenbedingungen sind hier im Optimum linear unabhängig).





Beispiel 5.6: Beispiel 5.2 nun mit Kuhn-Tucker

$(2(x_1 - 3), 2 (x_2 - 2)) + u_1 (2x_1, 2x_2) + u_2 (1, 2) + u_3 (-1, 0) + u_4 \ (0, -1) = (0, 0)$

$u_1 (x_1^2 + x_2^2 - 5) = 0$

$u_2 (x_1 + 2x_2 - 4) = 0$

$u_3 (- x_1) = 0$

$u_4 \ (- x_2) = 0$

$u_1, \ u_2, u_3, u_4 \geq 0$

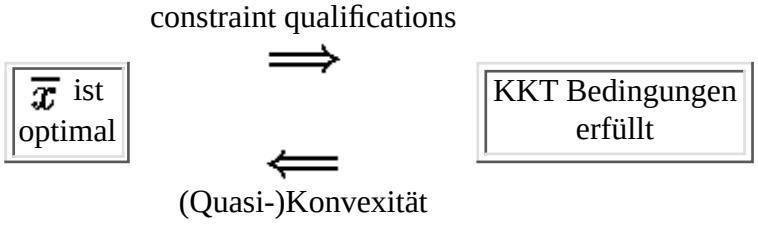
Man erhält also 6 Gleichungen (4 davon nichtlinear) mit 4 Variablen. man kann wieder Fallunterscheidungen anstellen (erhält 16 mögliche Fälle)
Erhalten den Punkt (2, 1)' . Durch Einsetzen erhalten wir dann wie früher $u_1 = 1/3$ und $u_2 = 2/3$, sodaß $u_1 , \ u_2 > 0$ und damit die Kuhn-Tucker Bedingungen
tatsächlich erfüllt sind.
Man stellt fest, daß (2, 1)' der einzige Kuhn-Tucker Punkt und damit bereits optimal ist.

Es gibt auch noch andere mögliche constraint qualifications, z.B. die Linearität der NB zu fordern.
Dann gilt $\overline{x}$ lokales Optimum $\Longrightarrow$ KKT Bedingungen erfüllt.

Wir haben bisher nur **notwendige** Optimalitätsbedingungen behandelt:
$\overline{x}$ lokales Optimum $\Longrightarrow$ Bedingungen erfüllt
Kann man auch **hinreichende** Optimalitätsbedingungen angeben?
Bedingungen erfüllt $\Longrightarrow \overline{x}$ lokales Optimum

In gewissen Situationen ist das möglich, etwa bei konvexen Optimierungsproblemen.

Theorem:
Es sei ein konvexes Optimierungsproblem gegeben, d.h. **F(x)** und alle $\mathbf{g_i (x)}$ sind konvex.
Sei $\overline{x}$ eine zulässig Lösung, für die die KKT Bedingungen erfüllt sind, dann ist $\overline{x}$ bereits die globale Lösung des Problems.



5.5. Gleichungen als Nebenbedingungen

Bisher haben wir nur Ungleichungen als Nebenbedingungen berücksichtigt, nun wollen wir auch Gleichungen dazu nehmen.

F(x) $\rightarrow$ min!
$g_1 \leq 0$
.
.
$g_m \leq 0$
$h_1 = 0$
.
.
$h_r = 0$

Wieder setzen wir differenzierbarkeit aller Funktionen im Punkt $\overline{x}$ voraus.

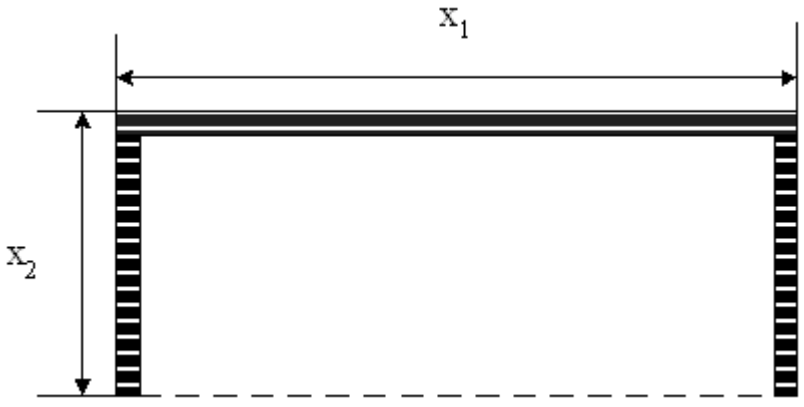
Theorem:
Es seien $\nabla g_i(\overline{x})$ (i $\in$ I) und $\nabla h_j(\overline{x})$ (j = 1, ... , r) linear unabhängig. Wenn $\overline{x}$ lokales Optimum ist, so gibt es Skalare u_i (i $\in$ I) und v_j (j = 1, ... , r) sodaß

(1) $(\nabla F(\overline{x}))' + \sum_{i \in I} u_i (\nabla g_i(\overline{x}))' + \sum_{j=1}^{r} v_i (\nabla h_j(\overline{x}))' = 0$
(2) $u_i \geq 0$ (i $\in$ I)

(Bem.: Alle Gleichheitsnebenbedingungen sind aktiv und die v_j sind nicht vorzeichenbeschränkt.)

Äquivalent dazu wären folgende Bedingungen:
(a) $(\nabla F(\overline{x}))' + \sum_{i=1}^{m} u_i (\nabla g_i(\overline{x}))' + \sum_{j=1}^{r} v_i (\nabla h_j(\overline{x}))' = 0$
(b) $u_i \cdot g_i(\overline{x}) = 0$ (i = 1, ..., m)
(c) $u_i \geq 0$ (i = 1, ..., m)
Zusätzlich müssen die Gleichheitsbedingungen erfüllt sein.

Beispiel 5.7: *(Wartehäuschen)*



Hinterwand höchstens 3m lang. Die Wände sollen insgesamt 4m lang sein. Die überdachte Fläche

Hinterwand höchstens 3m lang. Die Wände sollen insgesamt 4m lang sein. Die überdachte Fläche ist zu maximieren.

$x_1\, x_2 \rightarrow$ max !
$x_1 \leq 3$
$x_1 + 2x_2 = 4$

Also:
$F(x_1,x_2) = - x_1\, x_2 \rightarrow$ min !
$g_1(x_1,x_2) = x_1 - 3 \leq 0$
$h_1(x_1,x_2) = x_1 + 2x_2 - 4 = 0$

$\nabla F(x_1,x) = (-x_2 \ , -x_1)$
$\nabla g_1(x_1,x_2) = (1, 0)$
$\nabla h_1(x_1,x_2) = (1, 2)$

Kuhn-Tucker:

$$\begin{pmatrix} -x_2 \\ -x_1 \end{pmatrix} + u_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + v_1 \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$u_1 (x_1 - 3) = 0$
$u_1 \geq 0$
$x_1 + 2x_2 - 4 = 0$

2 Fälle unterscheiden:
1) $u_1 = 0$. Dann folgt daraus $x_2 = v_1$ und $x_1 = 2\, v_1 = 2\, x_2$ und nach Einsetzen in die letzte Gleichung $x_1 = 2$ und $x_2 = 1$

Der Punkt (2,1)' ist zulässiger KKT-Punkt mir dem Zielfunktionswert 2 m^2

1) $u_1 > 0$. Daraus folgt $x_1 = 3$ und $x_2 = 1/2$. Nachdem daraus folgt $u_1 = -1$ und $v_1 = 3/2$, ist (3, 1/2)' kein zulässiger KKT-Punkt.

Daraus folgt, daß der Punkt (2, 1)' schon das Optimum war.

Sonderfall m=0: Nur Gleichungen

Dann hat man nur noch

(1') $(\nabla F(\overline{x}))' + \sum_{j=1}^{r} v_i (\nabla h_j(\overline{x}))' = 0$

und die Gleichheitsbedingungen: $h_j(\overline{x}) = 0$ (j = 1, ..., r)

Diese beiden Gleichungen kann man folgendermaßen zusammenfassen:

$L(x,v) := F(x) + \sum_{j=1}^{r} v_j h_j(x)$ ("**Lagrange Funktion**")

Partielle Ableitungen von L nach **allen** Variablen $x_1, \ldots, x_n, v_1, \ldots, v_r$ bilden:

$\nabla F(x) + \sum_{j=1}^{r} v_j \nabla h_j(x)$ Ableitungen nach $x_1, \ldots, x_n$,

$h_1(x), \ldots, h_r(x)$ Ableitungen nach $v_1, \ldots, v_r$

Setze alle Ableitungen = 0. Dann liefert der erste Teil gerade die Gleichung (1') und der zweite Teil gerade die Gleichheitsbedingungen.

5.6. Straffunktionen und Barrierefunktionen

A) Straffunktionen (penalty functions)

Mit Hilfe von Straffunktionen wird versucht ein Problem mit Nebenbedingungen in ein Problem ohne Nebenbedingungen überzuführen. Das Verlassen des zulässigen Bereiches wird bestraft (die Nebenbedingungen werden in die Zielfunktion "eingebaut").

Beispiel:

Ursprüngliches Problem (P):
$F(x) \rightarrow \min!$
$h(x) = 0$
Transformiertes Problem (P')
$F(x) + \mu\,(h(x))^2 \rightarrow \min!$
$x \in \mathbf{R}^n$

$\mu\,(h(x))^2$ sind die "Strafkosten" für die Verletzung von Nebenbedingungen.

Wenn man μ hinreichend groß wählt, wird eine optimale Lösung von (P') einen Wert h(x) nahe bei 0 haben, da andernfalls eine hohe "Strafe" $\mu\,(h(x))^2$ anfallen würde.

Beispiel:

Ursprüngliches Problem (P):
$F(x) \rightarrow \min!$
$g(x) \leq 0$

Hier wäre der Strafterm $\mu\,(g(x))^2$ nicht günstig, da die Strafe auch bei $g(x) < 0$ anfallen würde. Man nimmt daher als Strafterm in diesem Fall $\mu \max(0, g(x))$, also

Transformiertes Problem (P')
$F(x) + \mu \max(0, g(x)) \rightarrow \min!$
$x \in \mathbf{R}^n$

Allgemeines Prinzip: Positive Strafkosten für nicht zulässige, keine Strafkosten für zulässige Punkte. Also allgemein:

Ursprüngliches Problem (P):
$F(x) \rightarrow \min!$
$g_i(x) \leq 0 \qquad (i = 1, ..., m)$
$h_j(x) = 0 \qquad (j = 1, ..., r)$

Transformiertes Problem (P')
$F(x) + \mu \,.\, \alpha(x) \rightarrow \min!$
$x \in \mathbf{R}^n$

Die Funktion $\alpha(x)$ wird folgendermaßen gebildet:

$\alpha(x) = \sum_{i=1}^{m} \ldots + \sum_{j=1}^{r} \ldots$

$\alpha(x) := \sum_{i=1} \varphi(g_i(x)) + \sum_{j=1} \psi(h_j(x))$

mit stetigen Funktionen φ und ψ, die folgende Bedingungen erfüllen:

$\varphi(y) = 0$ für $y \leq 0$ und $\varphi(y) > 0$ für $y > 0$
$\psi(y) = 0$ für $y = 0$ und $\psi(y) > 0$ für $y \neq 0$

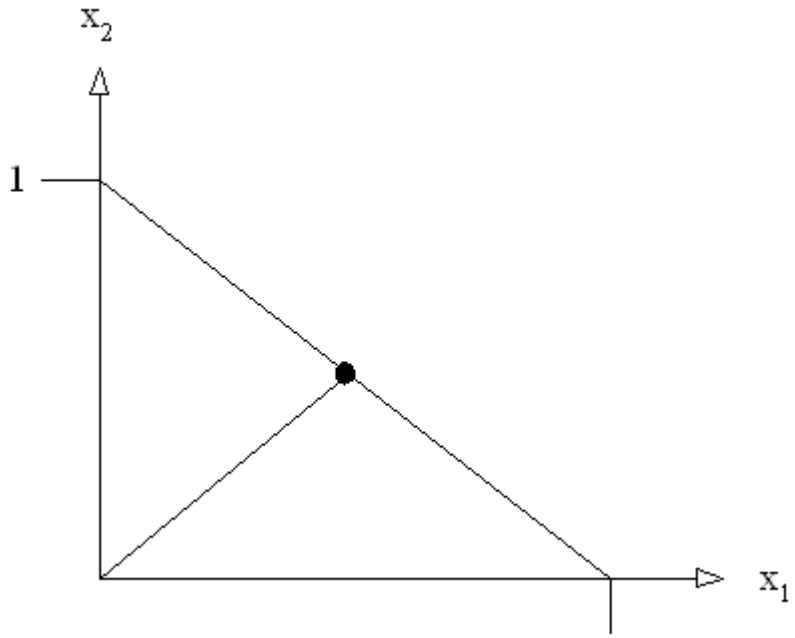
Typische Beispiele für solche Funktionen sind etwa:

$\varphi(y) = (\max(0, y))^p$
$\psi(y) = | y |^p$
(wobei p eine positive Zahl ist)

Die Straffunktion wäre dann:
$\alpha(x) = \sum_{i=1}^{m} [\max(0, g_i(x)]^p + \sum_{j=1}^{r} |h_j(x)|^p$

Die nun entstehende Funktion $F(x) + \mu \cdot \alpha(x)$ (mit großem μ) nennt man auch "Hilfsfunktion" und man verwendet sie nun als neue Zielfunktion (ohne Nebenbedingungen).

Beispiel:
Gesucht ist der Punkt der Geraden $x_1 + x_2 = 1$, der dem Ursprung am nächsten liegt.



Also:
$x_1^2 + x_2^2 \rightarrow$ min !
$x_1 + x_2 - 1 = 0$

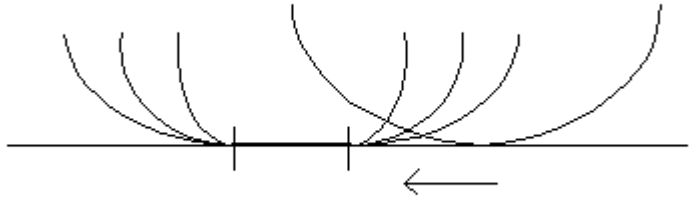
Optimaler Punkt ist (1/2, 1/2)

Mittels Methode der Straffunktionen:
Wir wählen p=2 und erhalten die Hilfsfunktion $F(x) = x_1^2 + x_2^2 + \mu (x_1 + x_2 - 1)^2$

Haben jetzt ein Problem ohne Nebenbedingungen. Also werden wir ableiten und = 0 setzen:

Ableiten nach x_1: $2 x_1 + 2 \mu (x_1 + x_2 - 1) = 0$
Ableiten nach x_2: $2 x_2 + 2 \mu (x_1 + x_2 - 1) = 0$

Also $x_1 = - \mu (x_1 + x_2 - 1)$ und $x_2 = - \mu (x_1 + x_2 - 1)$ und daher $x_1 = x_2$
Wenn wir nun in die erste Gleichung einsetzen, erhalten wir $(1 + 2 \mu) x_1 = \mu$ und damit $x_1 = x_2 = \mu / (1 + 2 \mu) = 1 / (2 + 1/\mu)$
Wählt man nun μ möglichst groß, damit die Strafe auch "zieht", dann bekommt man eine gute Näherung für den optimalen Wert $x_1 = x_2 = 1/2$.
Es ist allerdings zu beachten, daß die Lösungen der Hilfsprobleme $(1 / (2 + 1/\mu), 1 / (2 + 1/\mu))$ alle **nicht zulässig** sind im Bezug auf das ursprüngliche Problem (erst im Grenzwert $\mu \to \infty$).



Straffunktion

Je größer μ wird, umso härter wird eine Verletzung der Nebenbedingungen bestraft und umso näher wird man damit einer zulässigen Lösung kommen.
Ideal wäre eine Straffunktion folgender Art:
$\alpha(x) = 0$, falls $x \in \mathbf{S}$
$\alpha(x) = \infty$, falls $x \notin \mathbf{S}$
Mit so einer Funktion kann man aber wegen der Unstetigkeit nicht wirklich rechnen!

Straffunktionsmethode allgemein:

Sei das **ursprüngliches Problem** (P) gegeben (F, g_i, h_j stetige Funktionen) :

$F(x) \to$ min!
$g_i(x) \leq 0$ $\quad (i = 1, ..., m)$
$h_j(x) = 0$ $\quad (j = 1, ..., r)$

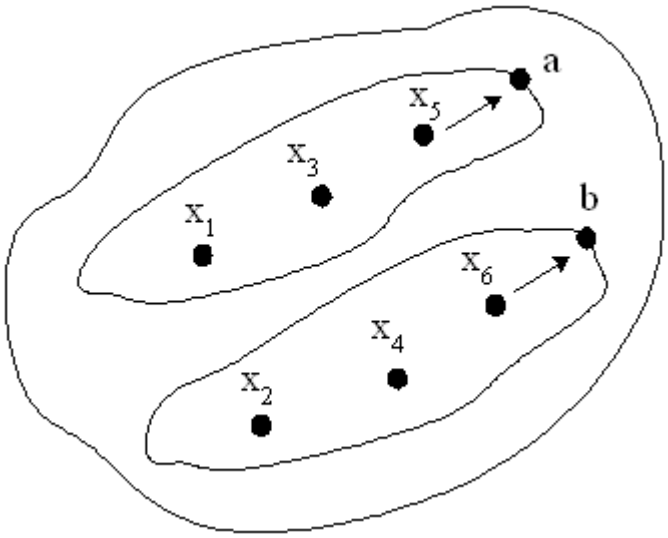
Transformiertes Problem (P') ($\alpha(x)$ definiert wie oben):

$F(x) + \mu \, . \, \alpha(x) \to$ min!
$x \in \mathbf{R}^n$

Wir lösen nun (P') und erhalten die Lösung x_μ und der Zielfunktionswert in x_μ ist $\theta(\mu) := F(x_\mu) + \mu . \alpha(x_\mu)$.

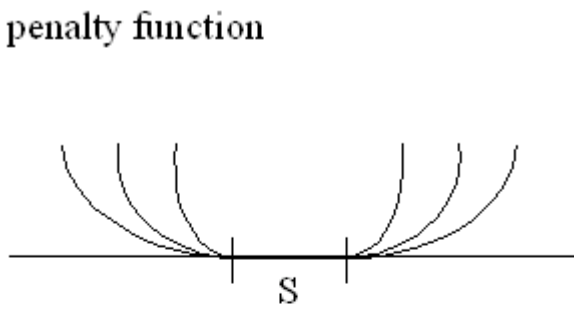
Theorem:
Falls das ursprüngliche Problem (P) lösbar ist und das Hilfsproblem (P') für jedes μ lösbar ist und

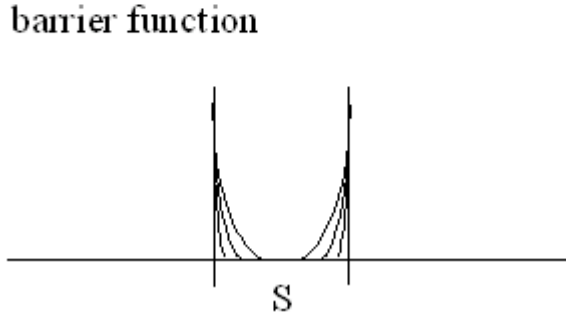
die Lösungen alle in einer beschränkten Teilmenge des $\mathbf{R}^n$ enthalten sind, so gilt:
(1) Der optimale Zielfunktionswert des ursprünglichen Problems (P) ist der Grenzwert $\lim_{\mu \to \infty} \theta(\mu)$.
(2) Einen optimalen Lösungspunkt des Problems erhält man, indem man den Grenzwert irgendeiner konvergenten Teilfolge von x_μ nimmt.



B) Barriere Funktionen (barrier functions)

Hier wird nicht erst die Verletzung der Nebenbedingungen bestraft, sondern sogar schon die Annäherung an die Grenze.





Barriere-Funktionen werden nur für Ungleichungs-Nebenbedingungen verwendet, also

$F(x) \rightarrow \min!$
$g_i(x) \leq 0 \qquad (i = 1, ..., m)$

Typische Barriere-Funktionen:

$B(x) = \sum_{i=1}^{m} \left(\frac{-1}{g_i(x)} \right)$ für $x \in \mathbf{S}$ und
$B(x) = \infty$, für $x \notin \mathbf{S}$

$-1 / g_i(x) > 0$ und wird umso größer, je mehr man sich der "Barriere" $g_i(x) = 0$ nähert.

Hilfsfunktion:
$F(x) + \mu . B(x) \rightarrow \min !$
(jetzt muß man μ gegen 0 gehen lassen)

Das Problem scheint nun nicht einfacher als das ursprüngliche zu sein, da man ja auch jetzt die Unterscheidung zwischen $x \in \mathbf{S}$ und $x \notin \mathbf{S}$ treffen muß. Praktisch aber ist es so, daß man, wenn man bei einem Startwert $x_0 \in \mathbf{S}$ beginnt, den zulässigen Bereich nie verläßt - dafür sorgt die "Barriere". Man braucht sich also um das "Außerhalb" gar nicht mehr zu kümmern.

Es gibt auch für die Barriere-Funktionen eine ähnlichen Konvergenzsatz wie bei den Penalty Funktionen.

6. Ganzzahlige Optimierung

6.1. Einleitung

Beispiel 6.1. *Proportionalwahlrecht*: Man teile die Mandate auf die kandidierenden Parteien so auf, daß die Mandatsanteile $M_1, ..., M_p$ den Stimmenanteilen möglichst genau entsprechen.
M Mandate insgesamt, p Parteien, Stimmenanteile $x_1, ..., x_p$ $(x_1 + ... + x_p = 1)$.
Zielfunktion: Die Summe der quadratischen Abweichungen minimieren.

$$\sum_{i=1}^{p} \left(\frac{M_i}{M} - x_i\right)^2 \rightarrow \min!$$
$$M_1 + \ldots + M_p = M$$

Die Lösung ist: $M_i = x_i M$ für $1 \leq i \leq p$
Im allgemeinen ist diese Lösung aber nicht ganzzahlig. Man kann aber nicht 76,342 Mandatare stellen. Es muß also irgendwie sicher gestellt werden, daß man zum Schluß ein ganzzahliges Ergebnis erhält.
Es sind komplizierte Verfahren notwendig (z.B. d'Hondtsches Verfahren) um die Ganzzahligkeit sicherzustellen.

Andere Beispiele:
Einsatzpläne für - Personal
- Maschinen
- Transportmittel
- Computer

Allgemeine Form eines ganzzahligen Optimierungsproblems:

$F(\mathbf{x}) \rightarrow \min !$
$g_i(\mathbf{x}) \leq 0 \quad (i = 1, ..., m)$
$\mathbf{x} \in \mathbb{Z}^n$

($\mathbb{Z}^n$ Menge der Vektoren mit ganzzahligen Koordinaten)

Bemerkung: Der Bereich des $\mathbf{R}^n$, für den $g_i(\mathbf{x}) \leq 0 \quad (i = 1, ..., m)$ gilt, kann unbeschränkt oder beschränkt sein. Im ersten Fall ist die zulässige Menge $S = \{\mathbf{x} \in \mathbb{Z}^n \mid g_i(\mathbf{x}) \leq 0 \quad (i = 1, ..., m)\}$ unendlich, im zweiten Fall ist sie endlich (könnte das dann auch der kombinatorischen Optimierung zuordnen).

Einen **Sonderfall** erhält man, wenn sowohl Zielfunktion als auch Nebenbedingungen **linear** sind (Linear ganzzahliges Problem, Integer Programming (IP)):

ZF: $\mathbf{c}^T\mathbf{x} \rightarrow \max!$
NB: $\mathbf{A}\,\mathbf{x} \leq \mathbf{b}$
Ganzzahligkeit $\mathbf{x} \in \mathbb{Z}^n$
(die x_i können vorzeichenbeschränkt sein oder nicht)

Besitzt das Problem eine spezielle Struktur, so kann auch ein Verfahren für kontinuierliche Probleme trotzdem immer ganzzahlige Lösungen ergeben (z.B. Simplexalgorithmus für das Zuordnungsproblem).
Im allgemeinen erhält man aber nichtganzzahlige Lösungen.

6.2. Lösen als kontinuierliches Problem

Man löst einfach das kontinuierliche Problem ($\mathbf{x} \in \mathbb{Z}^n$ weglassen) und sucht einen "Gitterpunkt" in der Nähe (Runden).

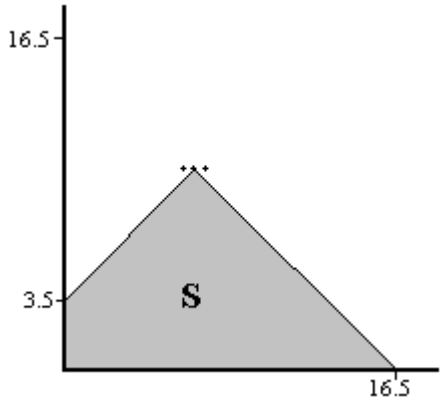
Dabei kann es allerdings zu Problemen kommen:

A) Die gerundete Lösung kann unzulässig sein

In welche Richtung soll man runden, um zulässig zu werden? Oft muß man mehrere Komponenten x_i simultan in geeigneter Weise runden, um in den zulässigen Bereich zu kommen.

Beispiel:

$x_2 \rightarrow$ max !
$-x_1 + x_2 \leq 3.5$
$x_1 + x_2 \leq 16.5$



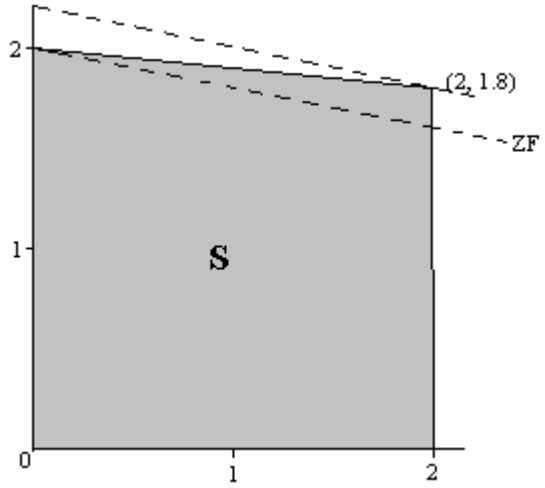
Der Simplexalgorithmus liefert als Lösung: $x_1 = 6.5$ und $x_2 = 10$
Aber weder durch Aufrunden von x_1 auf 7, noch durch Abrunden auf 6, erhält man eine zulässige Lösung. In diesem einfachen 2-dimensionalen Problem kann man wohl noch eine gute Lösung finden. Man kann sich aber wohl vorstellen um wieviel schwieriger das in hoch-dimensionalen Problemen werden kann.

B) Nicht optimale Lösung

Sogar, wenn man eine zulässige Lösung durch Runden gefunden hat, muß diese keineswegs optimal für das ganzzahlige Problem sein.

Beispiel:

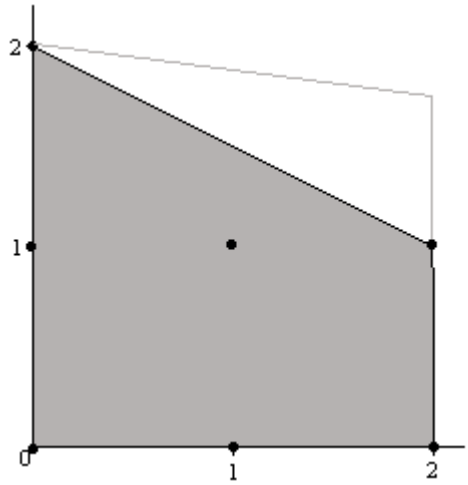
$x_1 + x_2 \rightarrow$ max !
$x_1 + 10x_2 \leq 20$
$x_1 \leq 2$
$x_1,\ x_2 \geq 0$
$x_1,\ x_2 \in \mathbb{Z}$



Der optimale Punkt des kontinuierlichen Problems ist (2, 1.8). Rundet man die 2. Komponente ab, erhältm man den zulässigen Punkt (2,1) mit Zielfunktionswert 7.
Der optimale Punkt des ganzzahligen Problems ist jedoch (0,2). Er liegt weit von dem gerundeten Punkt entfernt und hat einen viel besseren Zielfunktionswert 10.

C) Simplexalgorithmus für die konvexe Hülle

Um im Fall eines linearen ganzzahligen Problems mit dem Simplex-Algorithmus etwas ausrichten zu können, müßten wir die zulässige Menge einschränken, und zwar im Idealfall auf die "**konvexe Hülle**" der in der zulässigen Menge enthaltenen **ganzzahligen Gitterpunkte**. Im obigen Beispiel wäre das:



Der optimale Punkt der konvexen Hülle ist dann die Lösung des ganzzahligen Problems.

Praktisch ist es allerdings aussichtslos, die Nebenbedingungen, die diese konvexe Hülle definieren alle auf einmal finden zu wollen. Man geht daher **schrittweise** vor.
Dies ist auch der Grundgedanke der sogenannten **Schnittebenenverfahren**: Von der zulässigen Menge werden mit Hilfe geeigneter Geraden, Ebenen bzw. Hyperebenen immer wieder Stücke "weggeschnitten", bis man eine ganzzahlige Lösung erhält.

6.3. Schnittebenenverfahren von Gomory

Für **lineare** ganzzahlige Probleme.

Prinzip:

- Löse das entsprechende kontinuierliche Problem (man erhält i.A. eine nichtganzzahlige Lösung)
- Trenne die nichtganzzahlige Lösung mittels einer zusätzlichen linearen Nebenbedingung ab.
- Löse das neue Problem und wiederhole so lange, bis sich eine ganzzahlige Lösung ergibt.

Man löse also zunächst

$$\begin{array}{lcl} \mathbf{c}^T\mathbf{x} & \rightarrow & \text{max!} \\ \mathbf{A}\,\mathbf{x} & \leq & \mathbf{b} \\ \mathbf{x} & \geq & \mathbf{0} \end{array}$$

mit dem Simplex-Algorithmus. Das Endtableau besteht aus Gleichungen folgender Form (ohne ZF Zeile):

$$x_{ji} + \sum_{j \in I} a_{ij} x_j = b_i \quad \text{(i = 1, ..., m)}$$

wobei:
x_{ji} ..... Basisvariable der i-ten Zeile
I ..... Indexmenge der Nichtbasisvariablen
x_j ..... Entscheidungsvariablen + Schlupfvariablen

Setzen wir voraus, daß im ursprünglichen Problem alle a_{ji} und b_i ganzzahlig waren, dass müssen auch die Schlupfvariablen ganzzahlig sein.

Beispiel: *Endtableau von Beispiel 2.1.*

	x_1	x_2	x_3	x_4	x_5	r.S.
x_3	**0**	0	1	1/3	-1/3	2
x_2	**0**	1	0	1/2	0	6
x_1	**1**	**0**	**0**	**-1/3**	**1/3**	**2**

I = {4,5}

$x_3 + 1/3x_4 - 1/3\,x_5 = 2$ (j1 = 3)
$x_2 + 1/2x_4 + 0\,x_5 = 6$ (j2 = 2)
$x_1 - 1/3x_4 + 1/3\,x_5 = 2$ (j3 = 1)

Da alle b_i ganzzahlig sind, ist auch die Lösung ganzzahlig und wir sind fertig.
Nehmen nun an es gäbe eine Zeile mit Nummer h, für die b_h nicht ganzzahlig ist:

$x_{jh} + \sum_{j \in I} \alpha_{hj} x_j = b_h,$ für die $b_h \notin \mathbb{Z}$ (*)

α_{hj} und b_h werden nach unten gerundet:

$$\alpha_{hj} = \lfloor \alpha_{hj} \rfloor + r_{hj} \qquad 0 \leq r_{hj} < 1$$
$$b_h = \lfloor b_h \rfloor + r_h \qquad 0 < r_h < 1$$

damit läßt sich (*) dann folgendermaßen schreiben:

$$x_{jh} + \sum_{j \in I} \lfloor \alpha_{hj} \rfloor x_j + \sum_{j \in I} r_{hj} x_j = \lfloor b_h \rfloor + r_h$$
$$x_{jh} + \sum_{j \in I} \lfloor \alpha_{hj} \rfloor x_j - \lfloor b_h \rfloor = - \sum_{j \in I} r_{hj} x_j + r_h \qquad (**)$$

Für unsere angestrebte **ganzzahlige** Lösung sollen alle $x_j \in \mathbb{Z}$ sein, also ist die linke Seite von (**) ganzzahlig und damit auch die rechte.

$\sum_{j \in I} r_{hj} x_j$ ist sicher ≥ 0, also - $\sum_{j \in I} r_{hj} x_j \leq 0$.

Wegen $r_h < 1$ muß daher die rechte Seite von (**) < 1 sein, und da sie $\in \mathbb{Z}$ sein soll, kann das nur gehen, wenn sie ≤ 0 ist.

D.h.:

$- \sum_{j \in I} r_{hj} x_j + r_h \leq 0$ (***)

Man beachte, daß das eine **Verschärfung** des ursprünglichen LP ist. In (***) haben wir nämlich die Ganzzahligkeit der Lösung schon verwendet!

Tatsächlich erfüllt die optimale Lösung x* des ursprünglichen LP die Bedingung (***) **nicht**.

Für x* gilt nämlich:

$x_j^* = 0$ für alle $j \in I$ (alle Nichtbasisvariablen sind null)

Also wird (***) zu $r_h \leq 0$

und das stimmt nicht, da wir $r_h > 0$ vorausgesetzt hatten (b_h war ja nichtganzzahlig).

Wir führen also eine zusätzliche lineare Nebenbedingung ein (mit neuer Schlupfvariable formuliert):

$- \sum_{j \in I} r_{hj} x_j + x_{n+1} = - r_h$

Damit bekommt das LP eine Variable und eine Nebenbedingung mehr.

Wir lösen wieder.

Die neue Lösung ist dann entweder

- bereits ganzzahlig → STOP

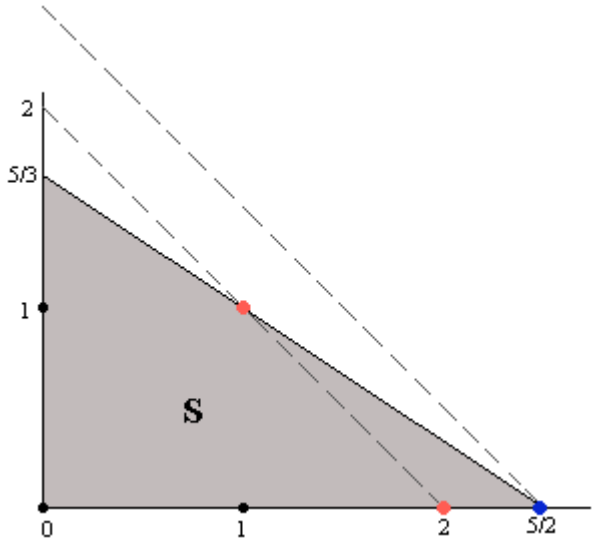
oder

- nicht ganzzahlig → Vorgang wiederholen.

Beispiel: Wir wollen auf einer Maschine mit einem Engpaß zwei Produkte 1 und 2 produzieren. Der Wert der produzierten Produkte ist gleich. Produkt 1 benötigt zwei Einheiten vom Engpaß, Produkt 2 drei Einheiten. Insgesamt stehen 5 Einheiten am Engpaß zur Verfügung.

Wieviel ist von 1 und 2 zu produzieren, wenn der Gesamtwert maximiert werden soll und aus produktionstechnischen Gründen nur ganze Einheiten von 1 und 2 produziert werden können?

$x_1 + x_2 \rightarrow$ max !
$2x_1 + 3x_2 \leq 5$
$x_1, x_2 \geq 0$
$x_1, x_2 \in \mathbb{Z}$



Zunächst läßt man $x_1, x_2 \in \mathbb{Z}$ weg. Das Starttableau ergibt sich zu:

	x_1	x_2	x_3	r.S.
Z	**-1**	-1	0	0
x_3	**2**	3	1	5

Daraus folgt:

	x_1	x_2	x_3	r.S.
Z	0	1/2	1/2	5/2
x_1	1	3/2	1/2	5/2

In der Zielfunktionszeile sind jetzt nur noch nichtnegative Koeffizienten, daher ist man mit dem Simplex fertig.

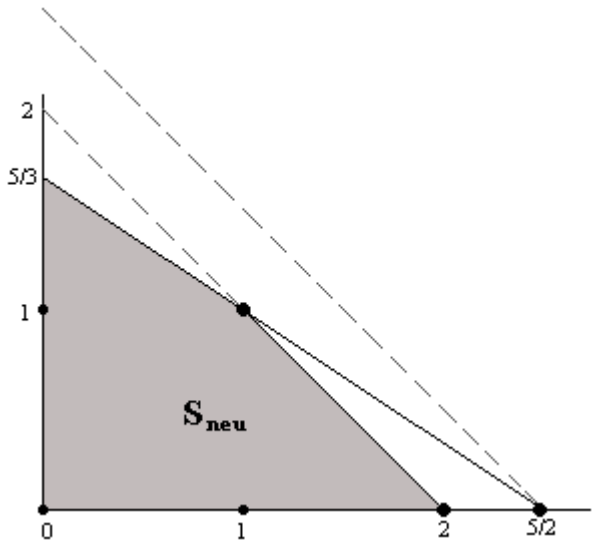
Lösung $x_1^* = 5/2, x_2^* = 0$ ist **nicht ganzzahlig** !

Auf der rechten Seite der ersten (und einzigen) Nebenbedingung steht ein Wert $b_{h \notin \mathbb{Z}}$ (h = 1)

Man setzt Zusatznebenbedinung an und schneidet eine Ecke ab! Dazu rundet man die Koeffizienten der h-ten Nebenbedingung nach unten ab:

$$\alpha_{12} = 3/2, \quad \lfloor \alpha_{12} \rfloor = 1, \quad r_{12} = 1/2$$
$$\alpha_{13} = 1/2, \quad \lfloor \alpha_{13} \rfloor = 0, \quad r_{13} = 1/2$$
$$b_1 = 5/2, \quad \lfloor b_1 \rfloor = 2, \quad r_1 = 1/2$$

$-(r_{12} x_2 + r_{13} x_3) + x_4 = - r_1$ (x_4 neue Schlupfvariable)
$-1/2 x_2 - 1/2 x_3 + x_4 = - 1/2$



Neues Tableau:

	x_1	x_2	x_3	x_4	r.S.
Z	0	1/2	1/2	0	5/2
x_1	1	3/2	1/2	0	5/2
x_4	0	-1/2	-1/2	1	-1/2

Nicht alle b_i sind ≥ 0!
Eigentlich müßten wir jetzt die Groß-M-Methode anwenden. Wenden hier einen Trick an und pivotieren nach einem negativen Element der letzten Zeile und hoffen, daß alle b_i positiv werden.
Wir nehmen das zweite Element der letzten Zeile und erhalten:

	x_1	x_2	x_3	x_4	r.S.
Z	0	0	0	1	2
x_1	1	0	-1	3	1
x_2	0	1	1	-2	1

Tatsächlich sind nun alle $b_i \geq 0$. Alle Koeffizienten der Zielfunktionszeile ≥ 0, also haben wir das Endtableau erreicht.
Es ist $x_1^* = 1$, $x_2^* = 1$. Das ist eine der beiden optimalen Lösungen. Und sie ist ganzzahlig, daher stoppt der Algorithmus.
Zielfunktionswert = 2.

Man kann mit dem Gomory Verfahren auch **gemischt-ganzzahlige** lineare Probleme lösen. Von gemischt-ganzzahligen Problemen spricht man, wenn nur für bestimmte Komponenten x_i des Vektors **x** Ganzzahligkeit gefordert wird, für andere aber nicht.

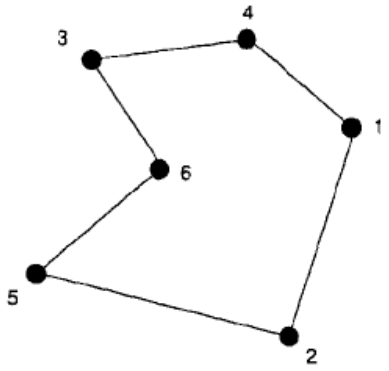
6.4. Kombinatorische Optimierung - Überblick

In der kombinatorischen Optimierung wird die zulässige Menge S stets als endlich vorausgesetzt. Ihre Elemente haben eine *kombinatorische* Struktur. Das können etwa sein:

- Permutationen
- Teilmengen
- Bäume

Permutationen:

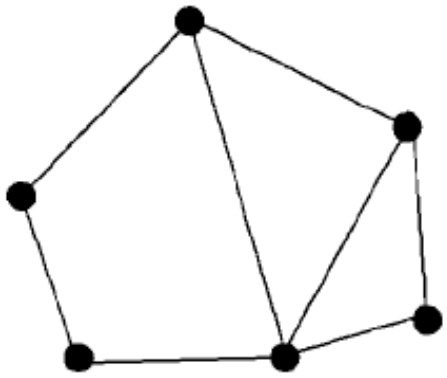
- Zuordnungsproblem (Assignemnt Problem)
  - Z.B.: Einer Menge von Mitarbeitern soll eine gleich große Menge von Aufgaben zugewiesen werden, sodass jeder Mitarbeiter genau eine Aufgabe bekommt und die Aufgaben bestmoglich erledigt werden.
- Rundreiseproblem (Travelling Salesman Problem)



  - Z.B.: In welcher Reihenfolge soll ein Vertreter n Städte besuchen, sodass die Länge der Tour minimal wird?
- Job-Shop Problem: n Jobs, m Maschinen; gesucht: optimale Reihenfolge, in der die Jobs an die Maschinen kommen.

Teilmengen:

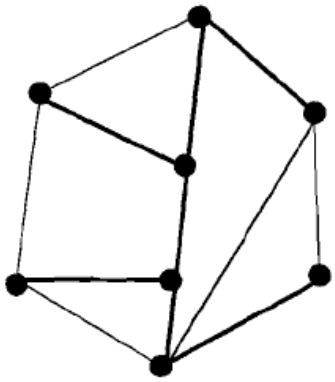
- Rucksackproblem (Knapsack Problem): Man wähle aus n Gegenständen einzelne Gegenstande so aus, dass ein bestimmtes Gewicht nicht überschritten und der Gesamtnutzen der ausgewählten Gegenstande maximiert wird.
- Maximum Clique Problem: Unter n Personen, die teilweise untereinander Kontakt haben, teilweise nicht, suche man eine maximale Teilmenge, in der jede Person mit jeder anderen Person Kontakt hat.



In diesem Graphen gibt es eine Clique der Größe 3, aber keine der Größe 4.

Bäume:

- Minimales Gerüst (Minimum Spanning Tree Problem)



  - Z.B.: In einem Kommunikationsnetz suche man ein Gerüst, das heißt eine Menge von Verbindungen, sodass jede Einheit mit jeder anderen kommunizieren kann. Die Gesamlänge soll minimal werden. Lösung ist immer ein Baum (kreisfreier zusammenhängender Graph).

Eine Lösungsstrategie: Branch-and-Bound Verfahren

Im Prinzip könnte man ein kombinatiorisches Problem so lösen, dass man für alle zulässigen Lösungen (es gibt ja nur endlich viele) die Zielfunktion bestimmt und das Minimum nimmt. Dies bezeichnet man als vollständige Enumeration. In der Praxis ist vollständige Enumeration aus Laufzeitgründen meist undurchführbar. Z.B.: Bei 10 Variablen mit je 100 möglichen Werten müsste man $100^{10} = 10^{20}$ Möglichkeiten durchprobieren.
Es ist daher in der Regel nötig, die Menge der zulässigen Lösungen, deren Zielfunktionswert man bestimmt, einzuschränken. Dazu eignet sich untenstehender Algorithmus.
Gegeben sei das kombinatorische Optimierungsproblem

$F(x) \rightarrow \min!$
$x \in \mathbf{S}$

Algorithmus. (Branch-and-Bound)

Liste := $\{S\}$; /* Liste der zu durchsuchenden Teilmengen */

$U := \infty$; /* U = *upper bound* */

solange Liste nicht leer {

  wähle eine Menge X aus Liste;

  entferne X aus Liste;

  branch: unterteile X in Teilmengen T_i;

  für jede Teilmenge T_i {

    bound: bestimme eine untere Schranke (*lower bound*) L_i für $F(x)$ in T_i;

wenn $L_i \geq U$, dann scheide T_i aus; /* hier suchen wir nicht weiter */

wenn $L_i < U$ und $x \in T_i$ mit $F(x) = L_i$ gefunden {

$x_{best} := x$;

$U := L_i$;

}

wenn $L_i < U$ und kein $x \in T_i$ mit $F(x) = L_i$ gefunden,

füge T_i der Liste hinzu; /* hier suchen wir weiter */

}

}

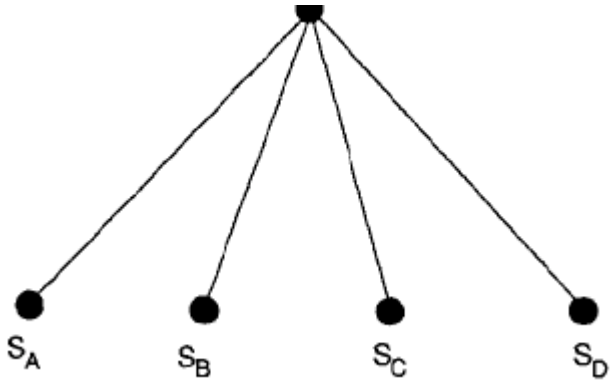
Die Teilmengen, die wegen $L_i \leq U$ ausgeschieden, das heißt nicht weiter untersucht werden, nennt man im Englischen *killed* oder *fathomed*. Dieser Algorithmus ist noch ziemlich allgemein. Man muß insbesonders spezifizieren:

1. die Strategie, nach der die Menge aus der Liste ausgewahlt und unterteilt wird (Schritt branch);
2. die Art, wie die untere Schranke bestimmt wird (Schritt bound).

Zunächst ein Beispiel:
Zuordnungsproblem mit Branch-and-Bound lösen. (Das Beispiel dient nur der Illustration des Branch-and-Bound-Verfahrens; für die Lösung des Zuordungsproblems gibt es effizientere spezifische Techniken.)
Kostentabelle:

	1	2	3	4
A	9	5	4	5
B	4	3	5	6
C	3	1	3	2
D	2	4	2	6

Möchte die Kosten minimieren.
Werden dabei die Struktur der Mengen und Teilmengen durch *Baum* darstellen.
Zulässige Lösung: Permutation von ABCD.
Zum Beispiel bedeutet CABD die Zuordnung: C $\rightarrow$ 1, A $\rightarrow$ 2, B $\rightarrow$ 3, D $\rightarrow$ 4.
S = Menge aller Permutationen von ABCD = {ABCD,ABDC, ... ,DCBA}
Zunächst enthält die Liste nur die Menge S. Also X = S aus Liste entfernen und in Teilmengen unterteilen:
S_A := {x $\in$ S: x beginnt mit A}
S_B := {x $\in$ S: x beginnt mit B}
S_C := {x $\in$ S: x beginnt mit C}
S_D := {x $\in$ S: x beginnt mit D}
In Baumdarstellung:



Damit ist der *Branch*-Schritt gemacht. Nun für jede Teilmenge den *Bound*-Schritt durchführen.
Fur S_A: Erste Zuordnung ist A $\rightarrow$ 1. Das verursacht Kosten der Höhe 9. Es bleiben 3 andere Zuordnungen. Untere Schranke L_1 der Kosten ist gesucht, also in jeder der 3 Spalten das Minimum nehmen, aber Zeile A bereits ausklammern. Ergibt:

$L_1 = 9+ 1+2+2 = 14.$

$L_1 < U$, aber kein $x \in T_1$ mit $F(x) = 14$ wurde gefunden (ACDC ist keine Permutation!). Also $T_1 = S_A$ an Liste dranhängen. In der Baumdarstellung: Der entsprechende Knoten *bleibt im Spiel*, von ihm aus kann also weiter verzweigt werden.
Fur S_B: Erste Zuordnung ist B $\rightarrow$ 1.

$L_2 = 4+1+2+2=9.$

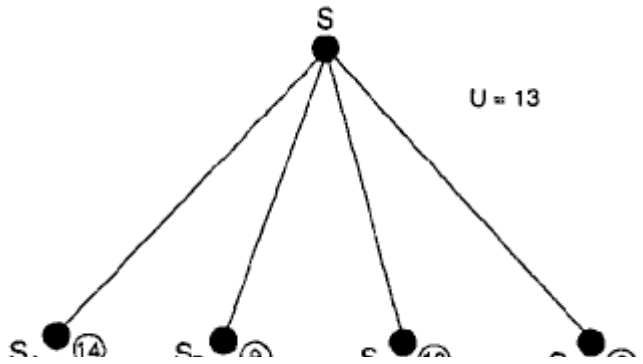
$L_2 < U$, kein $x \in T_2$ mit $F(x) = 9$ gefunden.
Fur S_C: Erste Zuordnung ist C $\rightarrow$ 1.

$L_3 = 3 + 3 +2+5= 13.$

$L_3 < U$, und L_3 wurde durch die Zuordnung CBDA realisiert!
Also x_{best} := CBDA, und U := 13.
Fur S_D: Erste Zuordnung ist D $\rightarrow$ 1.

$L_4 = 2+ 1+3+2 = 8.$

$L_4 < U$, kein $x \in T_4$ mit $F(x) = 8$ gefunden.

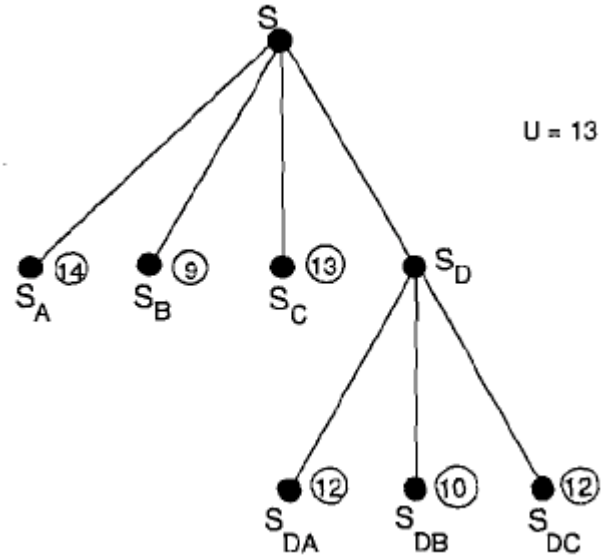


S_A ⑭ S_B ⑨ S_C ⑬ S_D ⑧

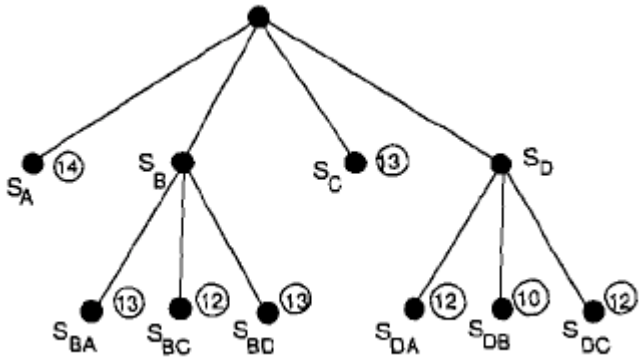
Liste enthält jetzt S_A, S_B, S_D.
X := S_D wählen: *lowest bound rule* (größte Erfolgschancen). Bei S_D weiter verzweigen.

S_{DA} := {x ∈ S: x beginnt mit DA}
S_{DB} := {x ∈ S: x beginnt mit DB}
S_{DC} := {x ∈ S: x beginnt mit DC}

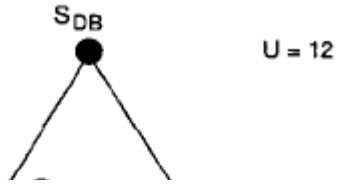
Vorgehen wie früher ergibt folgenden Baum:



Liste enthält jetzt S_A, S_B, S_{DA}, S_{DB}, S_{DC}
X := S_B wählen (lowest bound rule!)



Liste enthält S_A, S_{DA}, S_{DB}, S_{DC} · Die Knoten S_{BA} und S_{BD} wurden gekillt, S_{BC} ergab ein neues Optimum BCDA mit U = 12 und X := S_{DB}
Nur noch Zweig an S_{DB} herauszeichnen:



Hier liefert S_{DBA} ein neues Optimum DBAC mit U = 11. Liste enthält S_A, S_{DA}, S_{DC}· Da aber alle diese Mengen untere Schranken > 11 haben, werden sie alle in den folgenden Durchläufen gekillt. Optimale Lösung: DBAC mit Zielfunktionswert 11.

7. Metaheuristiken - Überblick

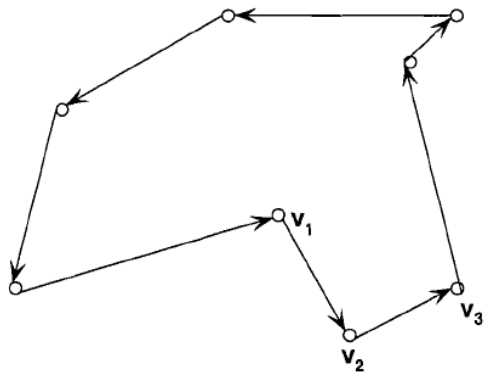
In der Praxis lassen sich nicht alle Optimierungs- und Suchprobleme innerhalb vernünftiger Laufzeit exakt lösen. Der folgende Abschnitt bietet eine Überblick über Verfahren zur näherungsweisen Lösung solcher Probleme, deren exakte Lösung zu aufwändig wäre, d.h. zur Bestimmung von guten, wenn auch nicht unbedingt optimalen Lösungen. Man nennt solche Verfahren Heuristiken. Ist eine Heuristik nicht bloß auf ein spezifisches Problem anwendbar, sondern (mit allfälligen Modifikationen) auf eine breite Klasse von Problemen, so spricht man von einer Metaheuristik. Mit Verfahren dieser Art werden wir uns im Folgenden beschäftigen.

7.1. Greedy-Heuristiken

Greedy-Algorithmen (Gierige Algorithmen) zeichnen sich dadurch aus, dass sie schrittweise denjenigen Folgezustand auswählen, der zum jeweiligen Zeitpunkt der Wahl das beste Ergebnis verspricht (siehe z.B. Gradientenverfahren in der Nichtlinearen Optimierung). Um unter den Folgezuständen eine Auswahl zu treffen, wird oft eine Bewertungsfunktion verwendet.
Der Greedy-Algorithmus garantiert das Auffinden der optimalen Lösung nur im Fall spezieller Probleme. Es ist naheliegend, den Greedy-Algorithmus in Fällen, wo er nicht zum Optimum führt, als Heuristik zu verwenden. In diesem Fall spricht man von einer Greedy-Heuristik. Man verwendet den Greedy-Algorithmus als lösungserzeugende Heuristik, eventuell werden danach lösungsverbessernde Heuristiken darauf angewendet.

Beispiel: Rundreiseproblem. Hier heisst *greedy*: möglichst kurze Distanzen wählen (gierig wird hier zu geizig).

Algorithmus. (Nächster Nachbar, Nearest Neighbor)
wähle einen Startknoten v_1;
für i = 1, ... ,n -1
    suche $v_{i+1} \notin \{v_1, \dots, v_i\}$, sodass $d(v_i, v_{i+1})$ minimal.



Sei L_{NN} die Länge der Tour, die Nächster Nachbar liefert, und L_{opt} die Länge der optimalen Tour.
Man kann beweisen:
Für Distanzen, die die Dreiecksungleichung erfüllen, gilt

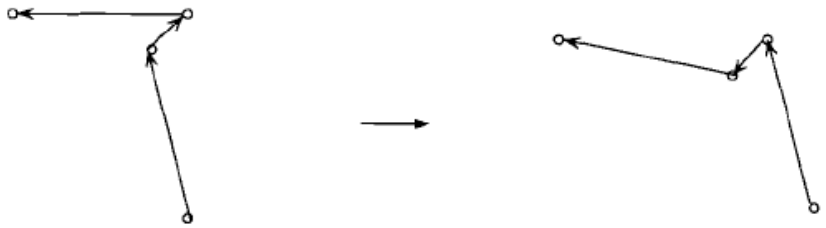
$$\frac{L_{NN}}{L_{opt}} \leq \lceil log_2 n \rceil / 2 + \frac{1}{2}$$

Zum Beispiel für $n = 100$:

$$\lceil log_2 100 \rceil / 2 + \frac{1}{2} = \frac{1}{2} * 7 + \frac{1}{2} = 4$$

Dies ist allerdings nur eine obere Schranke. Lösungsverbessernde Verfahren können außerdem die

Dies ist allerdings nur eine obere Schranke. Lösungsverbessernde Verfahren können außerdem die Länge der gefundenen Tour weiter reduzieren.
Lokale Verbesserungen lassen sich z.B. erreichen, indem man Teilstrecken der Länge k optimiert.



k = 4

Derselbe Anfangs-und Endknoten, dazwischen wurde verbessert.

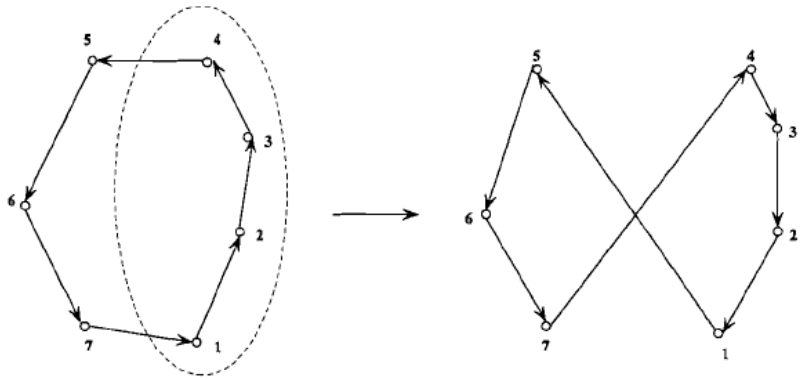
7.2 Lokale Suche (Local search)

Vorbereitung: Zu jeder zulässigen Lösung wird eine *Umgebung*, das heißt eine Menge benachbarter zulässiger Lösungen definiert. Im allgemeinen wählt man U(x) so, dass x ∈ U(x).
S zulässige Menge, x ∈ S.

$$x \mapsto U(x) = \{x_1, \ldots, x_m\} \ (x_i \in S)$$

Beispiel 1: Bei ganzzahliger Optimierung (die zulässige Menge ist eine Menge von Vektoren ganzer Zahlen) kann man eine Umgebung wie folgt definieren: Als benachbarten Vektor zu einem Vektor x fasst man jeden Vektor auf, der sich in jeder Komponente von x um höchstens 1 unterscheidet.

Beispiel 2: Beim Travelling Salesman Problem (TSP) kann man folgende Umgebung einer zulässigen Tour π definieren:
Die Touren π und π' sind benachbart, wenn π' aus π dadurch hervorgeht, dass man die Reihenfolge von k in π hintereinander besuchten Städten umdreht. Im Sonderfall k = 1 bleibt π unverändert.



Zu einer gegebenen Tour π gibt es i.a. sehr viele benachbarte Touren - aber immer noch viel weniger, als zulassige Touren insgesamt!

Algorithmus. (Iterierte Lokale Suche, Iterated Local Search)

$M := \infty$;

wähle Startlösung $x \in S$ zufällig;

bis Abbruchkriterium erfüllt {

  bestimme Umgebung $U(x) = \{x_1 \ldots, x_m\}$;

  wähle jenes $x_i =: x^*$ für das $F(x_i) = min$;

  wenn $F(x^*) < F(x)$ setze $x := x^*$;

  sonst /* lokales Minimum erreicht */ {

    wenn $F(x) < M$, setze $x_{best} := x$ und $M := F(x)$;

    wähle neuen Startwert $s \in S$ zufällig;

  }

}

Stoppt man sofort nach Erreichen eines lokalen Minimums, erhält man das einfache (nicht

Stoppt man sofort nach Erreichen eines lokalen Minimums, erhält man das einfache (nicht iterierte) Local-Search-Verfahren. Der wesentliche Unterschied des Local-Search-Verfahrens zum Greedy-Algorithmus liegt darin, dass die Lösung hier nicht schrittweise aufgebaut, sondern schrittweise geändert wird.

7.3 Simulated Annealing:

Simulated Annealing ist ein heuristisches Optimierungsverfahren, das eine Analogie zum *Annealing-Prozess* (Abkühlungsprozess) in der Festkörperphysik benutzt.
Wie beim Local Search nehmen wir an, dass eine *Umgebungsstruktur* festgelegt ist. Ebenfalls wie beim Local Search geht man pro Schritt grundsätzlich einmal von einer Lösung x zu einer benachbarten Lösung $y \in U(x)$.

(1) Ist y besser als x, macht man in jedem Fall bei y weiter.
(2) Ist y schlechter als x, macht man nur *mit einer gewissen Wahrscheinlichkeit* bei y weiter, mit der Gegenwahrscheinlichkeit bleibt man bei x.

Die Wahrscheinlichkeit, im zweiten Fall y zu akzeptieren, wählt man

- umso kleiner, je schlechter y im Vergleich mit x ist,
- umso kleiner, je niedriger der sogenannte Temperaturparameter ist.

Der Temperaturparameter c wird schrittweise abgesenkt.
. Die übliche Wahl der Wahrscheinlichkeit, y zu akzeptieren, ist folgende:

$$p_{x,c}(y) = exp\left(\frac{F(x) - F(y)}{c}\right)$$

Man beachte: F(x) -F(y) < 0, da ja y im Fall (2) schlechter als x ist. Also $p_{x,c}(y)$ umso kleiner, je größer der Unterschied zwischen F(x) und F(y).
Temperatur c niedrig $\Longrightarrow$ (F(x)-F(y))/c stark negativ, also $p_{x,c}(y)$ klein.

Algorithmus. (Simulated Annealing)

wähle Anfangswerte für x,c und L;
wiederhole, bis Abbruchkriterium erfüllt {
  für $l := 1$ bis L
  {
    wähle ein $y \in U(x)$;
    wenn $F(y) < F(x)$, dann setze $x := y$;
    sonst mit Wahrscheinlichkeit $p_{x,c}(y)$: setze $x := y$;
  }
  senke die Temperatur c;
  berechne L neu;
}

Außerdem: Die jeweils beste bisher gefundene Lösung x_{opt} speichern und aktualisieren!

Eine einfache Möglichkeit, c und L zu verändern, ist folgende:

c:= γ . c;
$L := \lceil \lambda \cdot L \rceil$;

wobei $0 < \gamma < 1$ und $\lambda \geq 1$ (z.E.: $\gamma = 0.95$, $\lambda = 1.1$). Man kann aber auch mit gleichbleibendem L arbeiten ($\lambda = 1$).
Ausführung des Schritts "mit Wahrscheinlichkeit $p_{x,c}(y)$: setze x := y;":
Man generiert eine Pseudo-Zufallszahl $Z \in [0, 1]$; wenn $Z \leq p_{x,c}(y)$, d.h.

$$Z \leq exp\left(\frac{F(x) - F(y)}{c}\right),$$

akzeptiert man den neuen Zustand y, sonst verwirft man ihn, d.h. bleibt bei x.

Im allgemeinen geht man von x aus bergab, mit gewissen (im Lauf der Zeit kleiner werdenden) Wahrscheinlichkeiten kann man sich aber auch bergauf bewegen. Das ermöglicht es einem, lokalen Minima wieder zu entkommen.
Es lässt sich beweisen: Unter gewissen Voraussetzungen über das Optimierungsproblem und über die Veränderungen von c und L, konvergiert die Wahrscheinlichkeit, dass der Simulated-Annealing-Algorithmus im n-ten Schritt beim Zustand x steht,

- gegen 0, falls x nicht gobales Optimum
- gegen 1 / (Anzahl der globalen Optima), falls x globales Optimum

Anders ausgedrückt: Simulated Annealing konvergiert gegen Gleichverteilung auf der Menge der globalen Optima. Aber: Es kann sehr lange dauern, bis man bei einem globalen Optimum angekommen ist. Daher sind aIle praktischen Implementierungen des Simulated Annealing als Näherungsalgorithmen (Heuristiken) anzusehen.

7.4 Genetische Algorithmen:

Genetische Algorithmen simulieren den Prozess der *natürlichen Evolution* beim Anpassen genetischer Merkmale an die Erfordernisse der Umwelt.
Es gibt viele Typen von genetischen Algorithmen, wir bringen hier den einfachsten.
Zulässige Lösung x binär codieren, als sogenanntes "Chromosom":
0100111010
Wir definieren zwei Typen von Operationen:

(a) Mutation: Nimm ein zufälliges Bit und flippe seinen Wert 0100111010 $\rightarrow$ 0101111010
(b) Crossover: Verwende zwei Chromosomen. Wähle eine zufallige Position zwischen 2 Bits, schneide beide Chromosomen an dieser Stelle auseinander und klebe die rechten Teile in vertauschter Reihenfolge wieder dran.

Die *fitness* $\phi(x)$ eines Chromosoms ist bei einem Maximierungsproblem als $(F(x))^{\beta}$ gegeben, bei einem Minimierungsproblem als $(F(x))^{-\beta}$, wobei β ein Parameter ist, den man geeignet wählt. (Experimentieren!) Im einfachsten Fall ist $\beta = 1$.

Algorithmus. (Genetischer Algorithmus)

erzeuge eine Generation 0 aus m Chromosomen mit Zufallsbits;
evaluiere jedes Chromosom x durch Berechnung der fitness $\phi(x)$;
wiederhole, bis Abbruchkriterium erfüllt {
  erzeuge die nächste Generation durch $m-$maliges zufälliges Auswählen eines
    Chromosoms, wobei die Auswahlwahrscheinlichkeit proportional zur
    fitness $\phi(x)$ ist ("Roulette-Rad-Regel");
  unterziehe eine gewisse Anzahl zufällig ausgewählter Chromosomen einer
    Mutation;
  unterziehe eine gewisse Anzahl zufällig ausgewählter Paare von Chromosomen
    einem Crossover;
  evaluiere jedes Chromosom der neuen Generation;
}

Auswahl gemäß der Roulette-Rad-Regel: Man dividiert zunachst die fitness-Werte jeweils durch deren Summe, um auf Wahrscheinlichkeiten zu kommen, und erstellt eine Tabelle der kumulierten Wahrscheinlichkeiten. Aus dieser lässt sich (ähnlich wie beim Simulated Annealing) eine Zufallsauswahl mit Hilfe einer zwischen 0 und 1 gleichverteilten Zufallszahl treffen.
Ein paar Varianten:ä

- Variante der Mutation: Jedes einzelne Bit mit einer bestimmten (kleinen) Wahrscheinlichkeit flippen, sodass sich auch mehrere Bits pro Chromosom ändern können.
- "Two-point-crossover": Zwei Schnittpunkte statt einem, Mittelteile vertauschen.
- "Elitismus": Die k besten Chromosomen werden automatisch in die neue Generation

übernommen.

E. Ant Colony Optimization:

Beispiel: Ein Vertreter löst sein TSP nach folgender Methode:

(1) Am 1. Tag beginnt er die Tour in Stadt 1 und wählt als nächste Stadt jeweils eine zufäIlige, noch nicht besuchte, wobei aIle noch nicht besuchten Städte zunächst gleiche Wahrscheinlichkeiten haben.
(2)Wenn er mit der Tour fertig ist, stellt er fest, wie gut sie ist. Ist sie besser als erwartet, erhöht er für jede an diesem Tag gewählte Teilstrecke i → j die Wahrscheinlichkeit, künftig von i aus gerade nach j zu fahren. Ist sie schlechter als erwartet, erniedrigt er für jede an diesem Tag gewählte Teilstrecke die entsprechende Wahrscheinlichkeit.
(3) An jedem der folgenden Tage wiederholt er die Schritte (1) - (2), wobei die Wahrscheinlichkeit für i → j gemäss den aktuellen Werten gewählt wird.

Das Verfahren lasst sich weiter verbessern:

- Nicht nur ein Vertreter ist unterwegs, sondern jeweils n Vertreter gleichzeitig. Sie verwenden gemeinsame Markierungen für die Teilstrecken i → j, d. h. jeder kann sich auch an den Erfahrungen der anderen orientieren.
- Die Wahrscheinlichkeit, zu einer bestimmten Nachbarstadt j zu fahren, wird (außer von den Markierungen) auch noch von den Distanzen d_{ij} abhängig gemacht.

  Genauer: p_{ij} proportional zu $\tau_{ij} \cdot \left(\frac{1}{d_{ij}}\right)^{\beta}$,

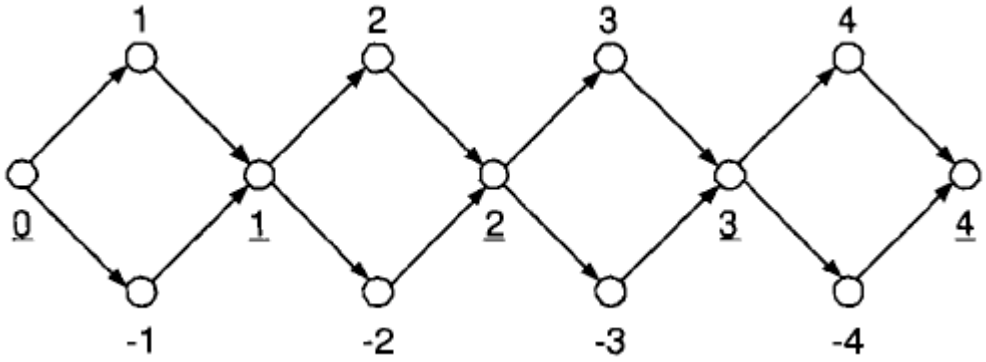
  wobei τ_{ij}, der sogenannte *Pheromon-Wert*, die Stärke der Markierung i → j angibt, und β ein frei einstellbarer Parameter ist. Damit $\sum p_{ij} = 1$, muss man

$$p_{ij} = \frac{\tau_{ij} \cdot \left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum_k \tau_{ik} \cdot \left(\frac{1}{d_{ik}}\right)^{\beta}}$$

  setzen, wobei die Summe über aIle noch nicht besuchten Städte k gebildet wird.

Bei Vorhandensein eines Parallelrechners kann die Tour jedes Vertreters auf einem eigenen Prozessor simuliert werden. Auf die sogenannte Pheromonmatrix (τ_{ij}) muss jedoch gemeinsam zugegriffen werden können. Die Sache funktioniert jedoch auch auf einem gewahnlichen PC, wenn man die Touren der einzelnen Vertreter hintereinander berechnet.
Der folgende Pseudocode bezieht sich auf die allgemeine Situation, d.h. die Anwendung von Ant Colony Optimization (ACO) auf ein beliebiges kombinatorisches Optimierungsproblem. (Auch Varianten für nicht-kombinatorische, z.B. kontinuierliche, Probleme existieren.) Man kodiert dazu eine Lösung in Form eines Pfades in einem sogenannten Konstruktionsgraphen. Für ein binäres Optimierungsproblem kann dieser etwa wie in der untenstehenden Zeichnung aussehen. Dabei wurde zur Illustration n = 4 gewahlt. Wahl von Knoten 1 bedeutet, dass Bit x_1 auf 1 gesetzt wird; Wahl von Knoten -1 bedeutet, dass das Bit x_1 auf 0 gesetzt wird, usf. Wie beim TSP sind die Pheromonwerte $\tau_{..}$ den Kanten (i,j) des Graphen zugeordnet.

ij



Algorithmus. (Ant Colony Optimization)

Initialisiere τ_{ij} auf den Kanten (i,j) des Konstruktionsgraphen;
für Iteration $n = 1, 2, \ldots$ {
  für Agent $s = 1, \ldots, S$ {
    positioniere Agent s im Startknoten des Konstruktionsgraphen;
    setze u, den bisher zurückgelegten Weg, gleich der leeren Liste;
    solange eine zulässige Fortsetzung des Wegs u existiert {
      wähle den Nachfolgerknoten j mit Wahrscheinlichkeit p_{ij}, wobei

        $p_{ij} = 0$, wenn die Fortsetzung (i,j) unzulässig ist, und
        $p_{ij} = (\tau_{ij} \cdot \eta(u)) \ / \ (\sum_{(i,k)} (\tau_{ik} \cdot \eta(u))$,
          wobei die Summe über alle zulässigen (i,k) geht, sonst;
      setze den aktuellen Weg u durch Hinzufügung von Kante (i,j) fort;
      setze $i := j$;
    }
  }
  multipliziere alle Pheromonwerte mit $1 - \rho$, wobei $\rho > 0$ ein Wert nahe bei 0;
  Verstärkung den besten Weg w^* unter den S gefundenen Wegen:
  addiere jeweils $\rho \cdot c$ zu den Pheromonwerten auf den Kanten von w^*;
}

"Zulässigkeit" der Fortsetzung des Wegs ist problemspezifisch definiert. Z.B. ist der Konstruktionsgraph bei einem TSP der vollständige Graph mit n Knoten, und ein Weg ist dann zulässig, wenn er keinen Knoten mehr als einmal besucht. Der sogenannte "Visibility-Wert" $\eta(u)$ kann hier gemäß einer beliebigen problemspezifischen Heuristik berechnet werden. Meist nimmt man dafür das Nutzen-Maß einer Greedy-Heuristik als Basis, wie auch im obigen TSP-Beispiel geschehen, wo sich $\eta(u)$ an der Nearest-Neighbor-Heuristik orientiert: $\eta(u) = d_{ij}^{-\beta}$. Man beachte, dass $\eta(u)$ vom gesamten bisher zurückgelegten Weg u abhängen darf (aber nicht muss).

7.5 Ant Colony Optimization:

Beispiel: Ein Vertreter löst sein TSP nach folgender Methode:

(1) Am 1. Tag beginnt er die Tour in Stadt 1 und wählt als nächste Stadt jeweils eine zufäIlige, noch nicht besuchte, wobei aIle noch nicht besuchten Städte zunächst gleiche Wahrscheinlichkeiten haben.
(2)Wenn er mit der Tour fertig ist, stellt er fest, wie gut sie ist. Ist sie besser als erwartet, erhöht er für jede an diesem Tag gewählte Teilstrecke i $\rightarrow$ j die Wahrscheinlichkeit, künftig von i aus gerade nach j zu fahren. Ist sie schlechter als erwartet, erniedrigt er für jede an diesem Tag gewählte Teilstrecke die entsprechende Wahrscheinlichkeit.
(3) An jedem der folgenden Tage wiederholt er die Schritte (1) - (2), wobei die Wahrscheinlichkeit für i $\rightarrow$ j gemäss den aktuellen Werten gewählt wird.

Das Verfahren lasst sich weiter verbessern:

- Nicht nur ein Vertreter ist unterwegs, sondern jeweils n Vertreter gleichzeitig. Sie verwenden gemeinsame Markierungen für die Teilstrecken i $\rightarrow$ j, d. h. jeder kann sich auch an den Erfahrungen der anderen orientieren.
- Die Wahrscheinlichkeit, zu einer bestimmten Nachbarstadt j zu fahren, wird (außer von den Markierungen) auch noch von den Distanzen d_{ij} abhängig gemacht.

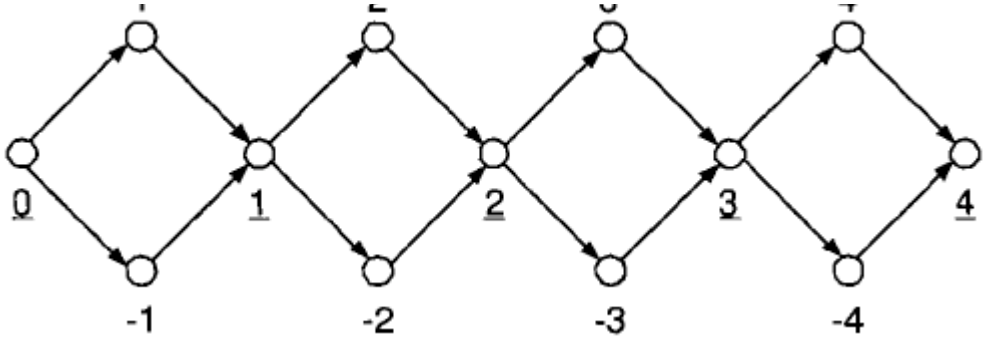
  Genauer: p_{ij} proportional zu $\tau_{ij} \cdot \left(\frac{1}{d_{ij}}\right)^{\beta}$,

  wobei τ_{ij}, der sogenannte *Pheromon-Wert*, die Stärke der Markierung i $\rightarrow$ j angibt, und β ein frei einstellbarer Parameter ist. Damit $\sum p_{ij} = 1$, muss man

$$p_{ij} = \frac{\tau_{ij} \cdot \left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum_k \tau_{ik} \cdot \left(\frac{1}{d_{ik}}\right)^{\beta}}$$

  setzen, wobei die Summe über aIle noch nicht besuchten Städte k gebildet wird.

Bei Vorhandensein eines Parallelrechners kann die Tour jedes Vertreters auf einem eigenen Prozessor simuliert werden. Auf die sogenannte Pheromonmatrix (τ_{ij}) muss jedoch gemeinsam zugegriffen werden können. Die Sache funktioniert jedoch auch auf einem gewahnlichen PC, wenn man die Touren der einzelnen Vertreter hintereinander berechnet.
Der folgende Pseudocode bezieht sich auf die allgemeine Situation, d.h. die Anwendung von Ant Colony Optimization (ACO) auf ein beliebiges kombinatorisches Optimierungsproblem. (Auch Varianten für nicht-kombinatorische, z.B. kontinuierliche, Probleme existieren.) Man kodiert dazu eine Lösung in Form eines Pfades in einem sogenannten Konstruktionsgraphen. Für ein binäres Optimierungsproblem kann dieser etwa wie in der untenstehenden Zeichnung aussehen. Dabei wurde zur Illustration n = 4 gewahlt. Wahl von Knoten 1 bedeutet, dass Bit x_1 auf 1 gesetzt wird; Wahl von Knoten -1 bedeutet, dass das Bit x_1 auf 0 gesetzt wird, usf. Wie beim TSP sind die Pheromonwerte τ_{ij} den Kanten (i,j) des Graphen zugeordnet.



Algorithmus. (Ant Colony Optimization)

Initialisiere τ_{ij} auf den Kanten (i,j) des Konstruktionsgraphen;
für Iteration $n = 1, 2, \ldots$ {
  für Agent $s = 1, \ldots, S$ {
    positioniere Agent s im Startknoten des Konstruktionsgraphen;
    setze u, den bisher zurückgelegten Weg, gleich der leeren Liste;
    solange eine zulässige Fortsetzung des Wegs u existiert {
      wähle den Nachfolgerknoten j mit Wahrscheinlichkeit p_{ij}, wobei

        $p_{ij} = 0$, wenn die Fortsetzung (i,j) unzulässig ist, und
        $p_{ij} = (\tau_{ij} \cdot \eta(u)) \ / \ (\sum_{(i,k)} (\tau_{ik} \cdot \eta(u))$,
          wobei die Summe über alle zulässigen (i,k) geht, sonst;
      setze den aktuellen Weg u durch Hinzufügung von Kante (i,j) fort;
      setze $i := j$;
    }
  }
  multipliziere alle Pheromonwerte mit $1 - \rho$, wobei $\rho > 0$ ein Wert nahe bei 0;
  Verstärkung den besten Weg w^* unter den S gefundenen Wegen:
  addiere jeweils $\rho \cdot c$ zu den Pheromonwerten auf den Kanten von w^*;
}

"Zulässigkeit" der Fortsetzung des Wegs ist problemspezifisch definiert. Z.B. ist der Konstruktionsgraph bei einem TSP der vollständige Graph mit n Knoten, und ein Weg ist dann zulässig, wenn er keinen Knoten mehr als einmal besucht. Der sogenannte "Visibility-Wert" $\eta(u)$ kann hier gemäß einer beliebigen problemspezifischen Heuristik berechnet werden. Meist nimmt man dafür das Nutzen-Maß einer Greedy-Heuristik als Basis, wie auch im obigen TSP-Beispiel geschehen, wo sich $\eta(u)$ an der Nearest-Neighbor-Heuristik orientiert: $\eta(u) = d_{ij}^{-\beta}$. Man beachte, dass $\eta(u)$ vom gesamten bisher zurückgelegten Weg u abhängen darf (aber nicht muss).

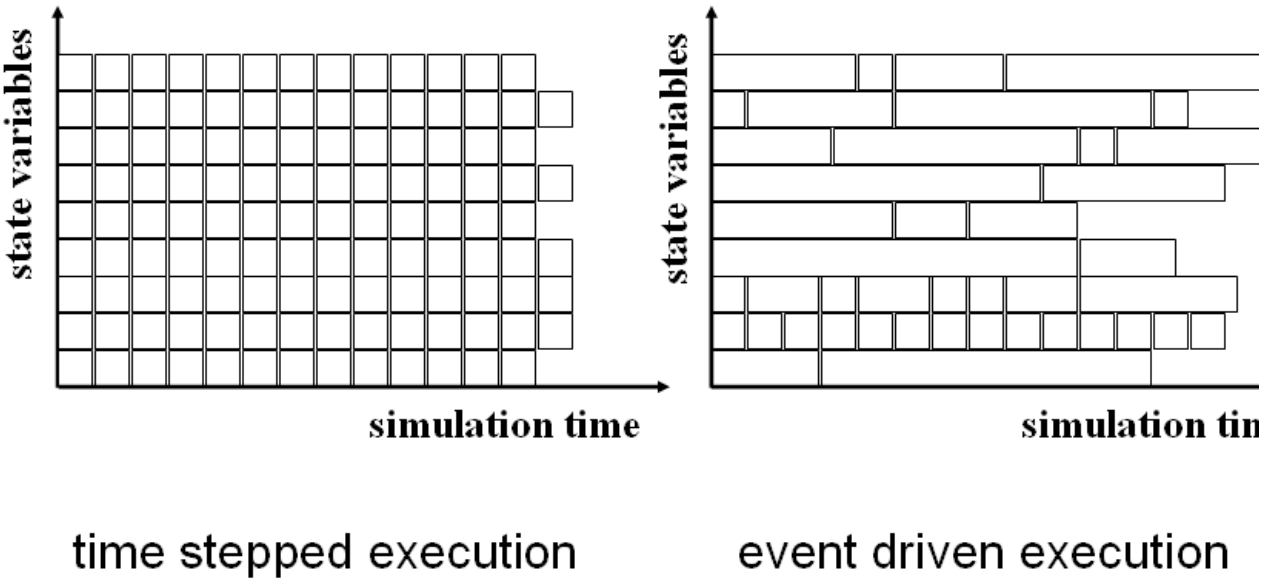
8. Simulation

8.1. Überblick

Simulation: Nachbilden eines Systems mit Hilfe von mathematischen Modellen, oft im Computer abgebildet und berechnet.
Anstelle von realen Objekten oder Vorgängen technischer, biologischer, ökonomischer oder ökologischer Art werden entsprechende mathematische oder physikalische Modelle untersucht - Simulation läßt sich daher auch verkürzt mit "Experiment am Modell" als Ersatz für ein tatsächliches Experiment charakterisieren (siehe Mattern, F.: Modellbildung und Simulation)

Ein System wird durch seine Zustandsvariablen beschrieben.
Bei dynamischen Modellen:

- kontinuierliche Zeit
  - Zustände verändern sich kontinuierlich in der Zeit.
  - Meist mit Hilfe von Differentialgleichungen modelliert.
  - typisch für
    - physikalisch-chemische Prozesse
    - komplexe Systeme, bei denen diskrete Abläufe vernachlässigt werden können (z.B. Bevölkerungsentwicklung)
    - längerfristige Zinsentwicklungen
- diskrete Zeit
  - Zustände verändern sich zu diskreten Zeitpunkten.
  - Zeitgesteuert: Simulationszeit ändert sich fortlaufend um konstantes Zeitinkrement
  - Ereignisgesteuert: Zustände verändern sich durch das Eintreten von Ereignissen
  - typisch für
    - Lagerhaltung
    - Telekommunikation
    - Warteschlangen



(http://www.cc.gatech.edu/classes/AY2003/cs4230_fall/Lectures.htm)

8.2. Monte Carlo Methoden

Modellierung zufälliger Prozesse

Wahrscheinlichkeitsdichten und -verteilungen

- **Gleichverteilung**: Ereignisse mit gleicher Wahrscheinlichkeit treten relativ häufig auf und dienen auch oft als Basis für alle weiteren Zufallsgrößen.
- **Diskrete Verteilungen**: Verteilungsfunktion ist Treppenfunktion
- **Normalverteilung**: Verteilung sehr vieler Größen in Natur, Technik und Ökonomie
- **Exponentialverteilung**: Etwa Ankunftsintervall von Personen oder Fahrzeugen an Bedienstationen oder Lebensdauer von Teilen ohne wesentliche Alterungsprozesse (gedächtnislos,d.h. bisherige Lebensdauer hat keinen Einfluss auf zukünftige Lebensdauer)
- **Gammaverteilung**: allgemeinere Form exponentieller Verteilungen (Exponentialverteilung ist eine Sonderform der Gammaverteilung)
- **Erlang-Verteilung**: auch Sonderfall der Gammaverteilung (z.B. kumulative Wartezeiten in Servicestationen)
- **Weibull-Verteilung**: für genauere Abbildung von Ausfallverhalten

Generierung von Zufallszahlen

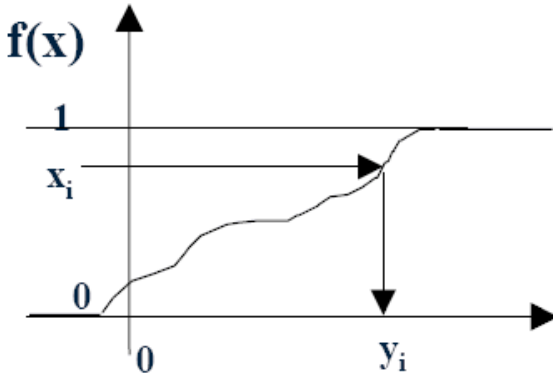
Fast alle Verteilungsfunktionen können von einer Gleichverteilung abgeleitet werden.
Erzeugung gleichverteilter Zufallszahlen ist die Basis für alle anderen Zufallszahlen.
Eigentlich werden im Computer *Pseudozufallszahlen* generiert (sind nicht wirklich zufällig, verhalten sich aber - im wesentlichen - so).
Es gibt viele verschiedene Verfahren, das bekannteste ist die Kongruenzmethode (nach Greenberger). Mit Hilfe der Berechnungsvorschrift

$x_{i+1} = (a * x_i + b) \bmod m$

mit m, dem Modulokoeffizienten (etwa $m=2^{50}$) und a dem Multiplikationskoeffizienten

Beliebige Gleichverteilungen erhält man aus der Gleichverteilung im Intervall [0,1] durch die Transformation
y = a + (b-a) * x, d.h. wenn x gleichverteilt in [0,1] ist, dann ist y gleichverteilt im Intervall [a,b]

Nichtlineare Verteilungsfunktionen erzeugt man über die Inversion der Verteilungsfunktion (falls diese explizit existiert):

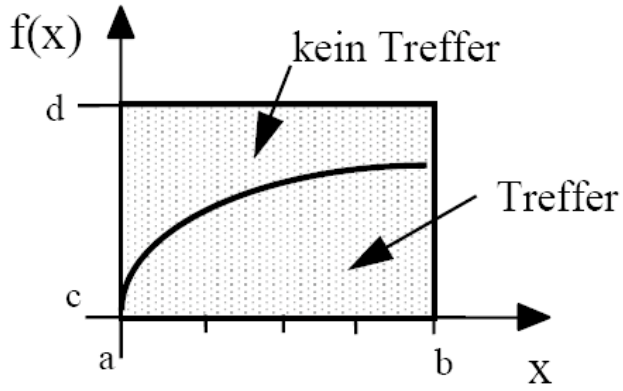


Setzt man in die Umkehrfunktion der Verteilungsfunktion F(x) eine im Intervall [0,1]

Setzt man in die Umkehrfunktion der Verteilungsfunktion F(x) eine im Intervall [0,1] gleichverteilte Zufallszahl x_i ein, so ergibt als Ergebnis eine Zufallszahl y_i, welche nach F(x) verteilt ist.
Das Prinzip heißt deshalb auch die Methode der Inversion der Verteilungsfunktion.

Monte-Carlo-Verfahren zur Berechnung von Integralen

- Integrale sind nicht immer analytisch lösbar.
- Eine Näherung des Fläächeninhalts kann mit der Monte-Carlo-Methode berechnet werden.
- Zur Lösung wird die gesuchte Fläche innerhalb einer einfach zu berechnenden Fläche (z.b. Quadrat oder Rechteck) eingeschlossen und für x und y werden entsprechend verteilte Zufallszahlen generiert (x im Intervall [a,b] und y aus [c,d]).



- Nach dem Einsetzen von x_i in f(x) ergibt sich eine Realisierung $f(x_i)$ und es wird getestet auf $y_i \leq f(x_i)$
- Falls die Ungleichung zutrifft, wird das Paar x_i , y_i als Treffer gewertet.
- Nach dem Theorem von Bernoulli, einer Variante des Gesetzes der großen Zahlen, konvergiert das Verhaltnis

$$\frac{\text{Anzahl der Treffer}}{\text{Anzahl der Versuche}} * \text{Fläche}$$

fur eine größere Anzahl von Versuchen gegen den gesuchten Flächeninhalt.
- Bei a=0, b=4, c=0 und d=1 ergibt sich eine Fläche von 4 Flächeneinheiten. Seien von 1000 Versuchen 562 als Treffer gezählt, so ergibt sich für den gesuchten Flächeninhalt A = 522 / 1000 * (4 * 1) = 2,88 .
- Zur Abschätzung der Genauigkeit des Verfahrens führt man M mal n-Versuche durch und ermittelt von den M-Einzelergebnissen das arithmetische Mittel und die Standardabweichung.

Beispiel:

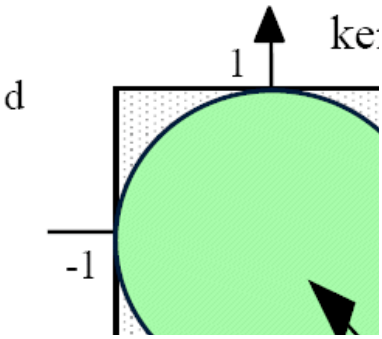
- Kreis wird bestimmt durch Kreisgleichung

$$x^2 + y^2 = r$$

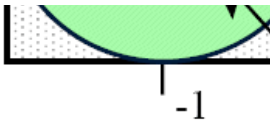
- Alle Wertekombinationen x_i, y_i mit

$$x_i^2 + y_i^2 \leq r$$

liegen auf oder innerhalb des Kreisbogens



liegen auf oder innerhalb des Kreisbogens.



- Mit einer Monte-Carlo-Simulation kann eine Näherung für die Fläche berechnet werden.

$$A_{MC} = \frac{\text{Treffer bei } (x_i^2 + y_i^2 \leq r)}{\text{Anzahl der Versuche}} * \text{Fläche des Rechtecks}$$

- Nach Gleichsetzung mit der analytischen Formel $A = \pi r^2$ kann z.B
  $\pi = A_{MC} / r^2 = \text{Treffer bei } (x_i^2 + y_i^2 \leq r) / \text{Versuc}$
  näherungsweise bestimmt werden.

Siehe Anwendung von Zufallszahlen bei der Monte-Carlo-Simulation (Thomas Wiedemann)

8.3. Diskrete Ereignis-Simulation

Veränderungen der Zustandsvariablen finden zu diskreten Zeitpunkten statt. Zwischen den einzelnen Zeitpunkten bleiben die Zustandsvariablen konstant.

- Zeitgesteuert (Time stepped): Simulationszeit ändert sich fortlaufend um konstantes Zeitinkrement
- Ereignisgesteuert (Event stepped): Zustände verändern sich durch das Eintreten von Ereignissen

Beispiel: Bedien- und Warteschlangen
Als Ausgangsgrößen werden die Ankunftsrate der zu bedienenden Objekte (Personen, Teile , Fahrzeuge etc.) und die Bedienrate (z.B. 20 Personen pro Stunde) verwendet.

Gesucht werden die mittlere Warteschlangenlänge, die mittlere Wartezeit pro zu bedienendem Objekt und bei stochastischen Einflüssen auch die maximale Warteschlangenlänge. Bei Bedarf werden auch verschiedene Strategien zur Organisation und Steuerung der Warteschlangen geprüft.

Grundlegende Bestandteile diskreter Simulationssysteme

Anstelle einer mathematischen Beschreibung erfolgt bei diskreten Simulationssystemen eine Systemnachbildung durch algorithmische Beschreibungen:

- Das Verhalten der Systemobjekte wird programmtechnisch nachgebildet.
- Eigenschaften der Systemobjekte werden auf Softwareobjekte mit entsprechenden Attributen übertragen.
- Der enge Zusammenhang zwischen Systemobjekt und Softwareobjekt war auch der Ausgangspunkt für die erste Realisierung objektorientierter Programmierung mit der Simulationssprache SIMULA bereits in den 60er Jahren.

Die Modellobjekte (Entities) lassen sich in folgende Modellbestandteile unterteilen :

- **Statische (permanente) Objekte**, häufig auch als Einrichtungen (oder treffender engl. Facilities) bezeichnet, dienen zur Nachbildung der Bedienstationen, Bearbeitungseinrichtungen oder sonstigen Systemobjekten mit Servicefunktionen.
- **Dynamische Objekte** (engl. auch Transactions) realisieren die Darstellung sich dynamisch durch das System bewegenden Personen, Fahrzeuge oder Teile.

Zeitsteuerung mit fixen Zeitinkrementen

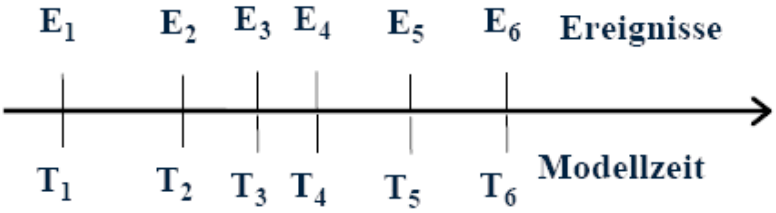
- In Analogie zur Schrittweite Δt bei der kontinuierlichen Simulation kann auch bei der diskreten Simulation mit einem vorgegebenen Zeitinkrement gearbeitet werden.
- **Zu jedem Zeitpunkt T_i** wird überpruft ob entsprechende Ereignisse eingetreten sind.

E_1 E_2 E_3 E_4 E_5 E_6 Ereignisse

- Die Realisierung des Simulationssteuerung ist bei diesem Verfahren relativ einfach, da nur die Modellzeit mit dem Inkrement monoton steigend hochgezählt wird.
- Weniger einfach ist die Modellierung von zufälligen Ereignissen, da bei jedem Takt das Eintreten des Ereignisses überpruft werden muß. Bei komplexeren Verteilungsfunktionen muß dabei die Vergangenheit berücksichtigt werden, was zu einem betrachtlichen Aufwand in der Modellbeschreibung führt.
- Das Verfahren der fixen Zeitinkremente wird nur selten und meist nur bei spezialisierten, hoch aggregierten Modellen verwendet, bei denen wenige Zustandsvariable sich sehr häufig ändern.

Zeitsteuerung mit variablen Zeitinkrementen

- Die einzelnen Zeitpunkte T_i können auch durch Auswertung des Modells ermittelt werden.
- Die Modellzeit wird damit nicht mehr durch die modellexterne Simulationssteuerung vorgegeben, sondern wird asynchron durch die Abläufe im Modell gesetzt.
- Ein Modellzeitpunkt entspricht mindestens einem Ereignis !



- Bei der **ereignisorientierten** Simulation wird durch die Simulationssteuerung ein **Systemkalender** (engl. Future event list) mit den nächsten möglichen Ereignissen geführt.
- Durch Bestimmung des Minimums aller anstehenden Ereigniszeitpunkte wird der nächste Modellzeitpunkt bestimmt und die Simulationsuhr entsprechend gesetzt.

Siehe Simulation von diskreten Modellen (Thomas Wiedemann)