Freie Universität Berlin

Seminar über Algorithmen

Prof. Dr. Helmut Alt

Lineare Programmierung Teil I

Lena Schlipf, Benjamin Jankovic

Freie Universität Berlin

Struktur des Vortrags

1. Was ist lineare Programmierung?
2. Motivation
3. Einführendes Beispiel
4. Formalisierung
5. Graphische Interpretation
6. Simplexalgorithmus

Was ist lineare Programmierung?

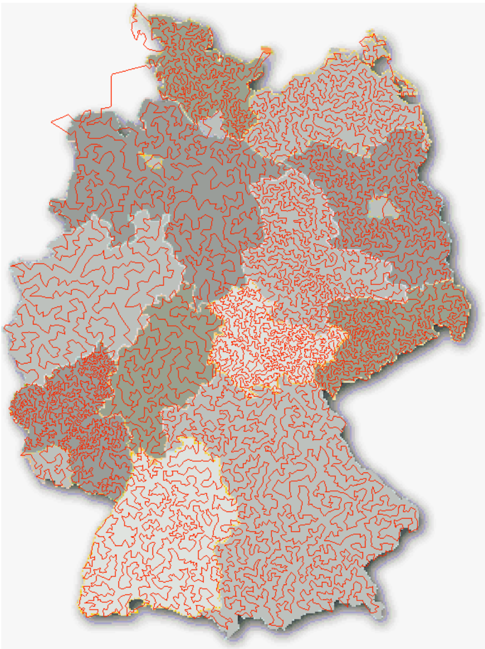
- Lineare Funktion mit n Variablen soll minimiert oder maximiert werden
- Für die Variablen müssen dabei gewisse Nebenbedingungen erfüllt sein

Motivation

Sehr viele Anwendungsformen:

- Standardverfahren, das z.B. bei der Planung von Produktionsabläufen zum Einsatz kommt
- Viele Optimierungsprobleme können formal als lineares Programm dargestellt werden
- Momentan beste Lösung des TSP-Problems beruht auf linearer Programmierung

TSP-Problem



Tour durch 15112 Städte in Deutschland

Weiterentwicklung der cutting-planes-Methode, die wiederum auf der linearen Programmierung beruht

2001:

David Applegate, Robert Bixby, Vašek Chvátal und William Cook

Einführendes Beispiel

Szenario: Wahlkampfpolitik

- Eine Politikerin möchte in ihrem Wahlkreis gewinnen. Dazu benötigt sie in mindestens 50% aller Stimmen in jedem der 3 Gebiete:
  - Stadt (100.000 Einwohner)
  - Vorstadt (200.000 Einwohner)
  - Land (50.000 Einwohner)

Einführendes Beispiel (2)

Dieses möchte sie durch geschickten Wahlkampf erreichen.
Dank sehr guter psychologischer Studien weiß sie:

Je 1000€ Werbeausgaben für ein bestimmtes Wahlkampfthema haben eine gewisse Anzahl Wählerstimmen zur Folge (in Tausend):

Wahlkampfthema	Stadt	Vorstadt	Land
Straßenbau	-2	5	3
Sicherheit	8	2	-5
Landwirtschaftsbeihilfe	0	0	10
Mineralölsteuer	10	0	-2

Problem

Wie erreicht die Politikerin ihr Ziel mit möglichst wenig Ausgaben?

- Möglichkeit:
  „Trial & Error“
- Aber:
  Woher weiß man, ob eine gefundene Lösung optimal ist?
- Darum:
  Formuliere Problem zunächst mathematisch

Mathematische Formulierung

- Einführung von 4 Variablen:
  Ausgaben in 1000€ für Wahlkampfthema:
  x_1: Kosten für Straßenbau
  x_2: Kosten für Sicherheitspolitik
  x_3: Kosten für Landwirtschaftsbeihilfe
  x_4: Kosten für Mineralölsteuer

- Die Forderung, daß z.B. in der Stadt 50%, also 50.000 der Wählerstimmen erlangt werden, sähe dann so aus:

$$-2x_1 + 8x_2 + 0x_3 + 10x_4 \geq 50$$

Mathematische Formulierung

- genügend Stimmen aus der Vorstadt werden gewonnen:
  $5x_1 + 2x_2 + 0x_3 + 0x_4 \geq 100$
- genügend Stimmen vom Land werden gewonnen:
  $3x_1 - 5x_2 + 10x_3 - 2x_4 \geq 25$
- Da die Kosten minimiert werden sollen,
  soll der Term $x_1 + x_2 + x_3 + x_4$
  minimiert werden
- Da es keine negativen Kosten gibt, gilt außerdem:
  $x_1, x_2, x_3, x_4 \geq 0$

Fertiges Lineares Programm

Das lineare Programm sieht also wie folgt aus:

Minimiere

$$x_1 + x_2 + x_3 + x_4$$

unter den Nebenbedingungen:

$$\begin{aligned}
-2x_1 + 8x_2 + 0x_3 + 10x_4 &\geq 50 \\
5x_1 + 2x_2 + 0x_3 + 0x_4 &\geq 100 \\
3x_1 - 5x_2 + 10x_3 - 2x_4 &\geq 25 \\
x_1, x_2, x_3, x_4 &\geq 0
\end{aligned}$$

Definition: lineares Programm

Optimiere lineare Funktion

$$f(x_1, x_2, \ldots, x_n) = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = \sum_{j=1}^{n} a_j x_j$$

mit linearen Nebenbedingungen

$$f(x_1, x_2, \ldots, x_n) = b$$

oder

$$f(x_1, x_2, \ldots, x_n) \leq b$$

oder

$$f(x_1, x_2, \ldots, x_n) \geq b$$

Darstellung linearer Programme

Jedes lineare Programm lässt sich in die beiden folgenden Darstellungen umformen:

- Die **Standardform**
- Die **Schlupfform** (oder auch Slackform)

Standardform

Wir wollen n reelle Zahlen bestimmen, welche die

lineare Funktion $\sum_{j=1}^{n} c_j x_j$
maximieren

Unter den Nebenbedingungen

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad i = 1,2,\ldots,m$$

$$x_j \ge 0 \qquad j = 1,2,\ldots,n$$

Standardform

Wir wollen n reelle Zahlen bestimmen, die die

lineare Funktion $\sum_{j=1}^{n} c_j x_j$ Zielfunktion

maximieren

Unter den Nebenbedingungen

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i = 1,2,\ldots,m$$

Nichtnegativitätsbedingungen

$$x_j \geq 0 \qquad j = 1,2,\ldots,n$$

Begriffe

Zulässige Lösung:
Variablenbelegung, die alle Nebenbedingungen erfüllt

Unzulässige Lösung:
Alle anderen Variablenbelegungen

Unlösbar:
Es gibt keine zulässige Lösung für ein lineares Programm

Zielfunktionswert:
Wert, den die Zielfunktion bei einer bestimmten Variablenbelegung annimmt

Optimale Lösung:
Zulässige Lösung mit optimalem, d.h. maximalem Zielfunktionswert

Unbeschränktes Programm:
Zulässige Lösungen existieren, aber keine optimale Lösung

Äquivalenz linearer Programme

- Zwei lineare Maximierungs-Programme heißen **äquivalent**, wenn die jeweilige Menge der zulässigen Zielfunktionswerte äquivalent ist.
- Ein lineares Maximierungs-Programm L und ein lineares Minimierungsprogramm L‘ heißen **äquivalent**, wenn jedem z aus der Menge der zulässigen Zielfunktionswerte von L ein -z aus der Menge der zulässigen Zielfunktionswerte von L‘ entspricht.

Umformen in Standardform

Probleme, die beim Umformen eines beliebigen linearen Programmes in ein äquivalentes Programm in Standardform auftreten können:

- Zielfunktion soll minimiert anstatt maximiert werden
- Es gibt Variablen ohne Nichtnegativitätsbedingung
- Es gibt Nebenbedingungen, die eine Gleichung sind
- Es gibt Nebenbedingungen, die eine Ungleichung sind, die aber „≥“ statt „≤“ fordern

Lösung der Probleme

- Minimierung ⇒ Maximierung

  Negiere alle Koeffizienten der Zielfunktion
- Nichtnegativitätsbedingung erfüllen

  Führe für x_i zwei neue Variablen $x_i' \geq 0$ und $x_i'' \geq 0$ ein und substituiere alle x_i mit $x_i' - x_i''$
- Gleichung ⇒ Ungleichung

  Ersetze A = B durch A ≤ B und A ≥ B
- ≥-Relation ⇒ ≤-Relation

  Multipliziere beide Seiten mit (-1)

Schlupfform

Maximierung einer linearer Funktion unter Beachtung linearer Gleichungen und Nichtnegativitätsbedingungen

Umformen:

- Forme beliebiges Programm in Standardform um
- Ersetze Ungleichungen durch Gleichungen

$$\sum_{i=1}^{n} a_{ij} x_j \leq b_i \quad \Rightarrow \quad x_{n+i} = b_i - \sum_{i=1}^{n} a_{ij} x_j, \quad x_{n+i} \geq 0$$

x_{n+i} ist eine sogenannte **Schlupfvariable**

Beispiel für eine Schlupfform

maximiere $2x_1 - 3x_2 + 3x_3$

unter den Nebenbedingungen:

$$
\begin{aligned}
x_4 &= 7 - x_1 - x_2 + x_3 \\
x_5 &= -7 + x_1 + x_2 - x_3 \\
x_6 &= 4 - x_1 + 2x_2 - 2x_3
\end{aligned}
$$

$\mathbf{x_1, x_2, x_3, x_4 \geq 0}$

Vereinfachte Schreibweise

$$
\begin{aligned}
z &= 2x_1 - 3x_2 + 3x_3 \\
x_4 &= 7 - x_1 - x_2 + x_3 \\
x_5 &= -7 + x_1 + x_2 - x_3 \\
x_6 &= 4 - x_1 + 2x_2 - 2x_3
\end{aligned}
$$

Die Nichtnegativitätsbedingungen aller Variablen und die Tatsache, daß z maximiert werden soll, werden hierbei nicht explizit genannt.

Die Variablen auf der linken Seite nennt man **Basisvariablen**, diejenigen auf der rechten Seite **Nichtbasisvariablen.**

Graphische Lösung des Problems

Weiteres Beispiel:

Düngemittelfabrik stellt zwei Düngemittel A und B aus den Ausgangsstoffen C,D und E her.

Verfügbarkeit der Ausgangsstoffe:

C: 1500t, D: 1200t, E: 500t

Chemische Zusammensetzung:

1t A = 2t C + 1t D +1t E

1t B = 1t C + 1t D

Gewinne pro Tonne:

A: 30€

B: 20€

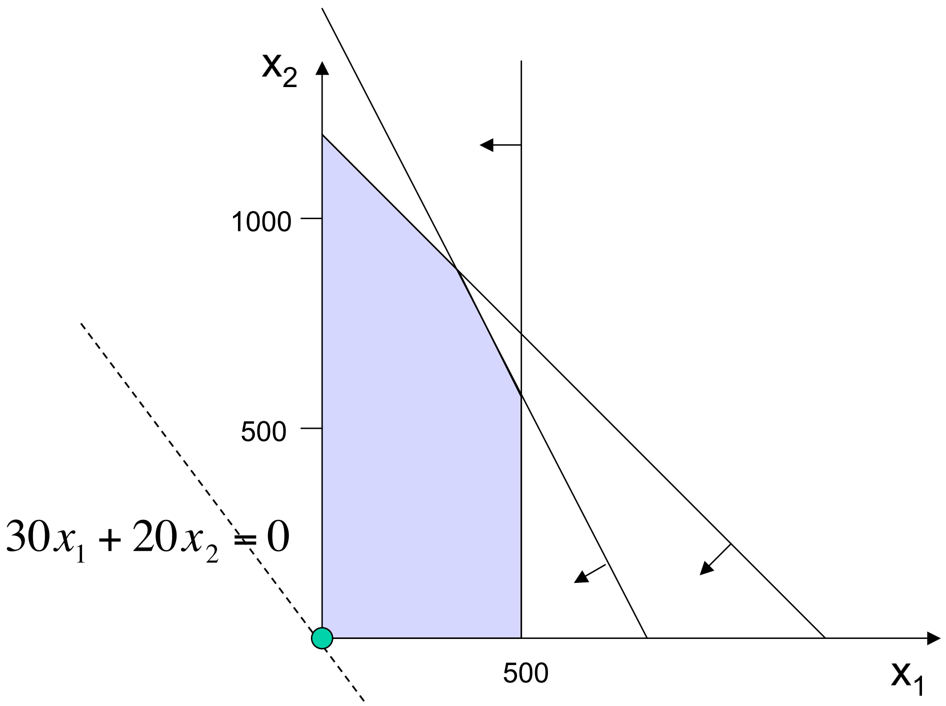
Maximiere:

$$30x_1 + 20x_2$$

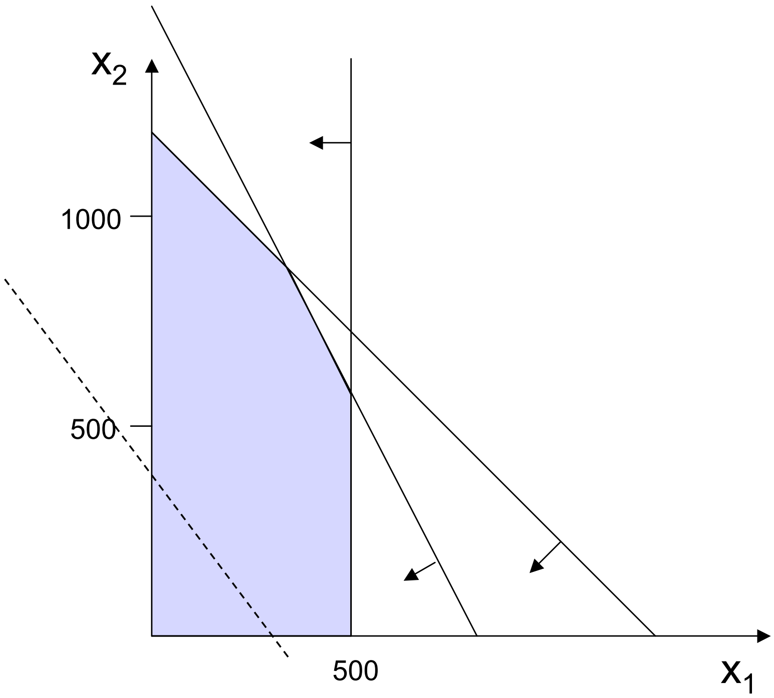
Nebenbedingungen:

$$\begin{aligned} 2x_1 + x_2 &\leq 1500 \\ x_1 + x_2 &\leq 1200 \\ x_1 &\leq 500 \\ x_1, x_2 &\geq 0 \end{aligned}$$

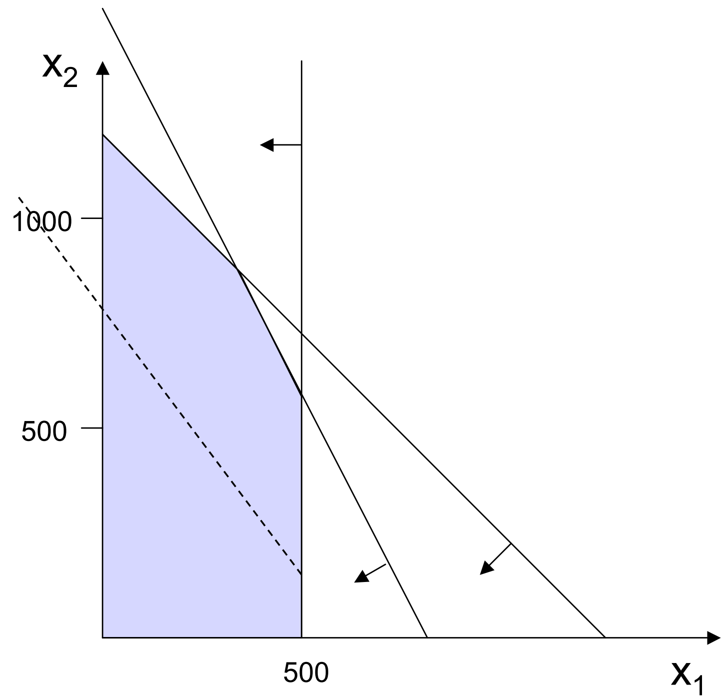
Graphisches Beispiel (1)



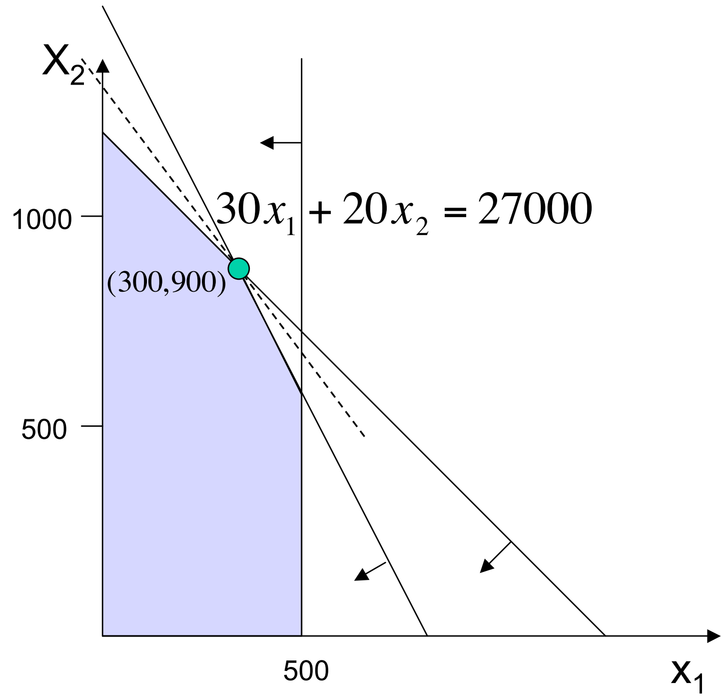
Graphisches Beispiel (2)



Graphisches Beispiel (3)

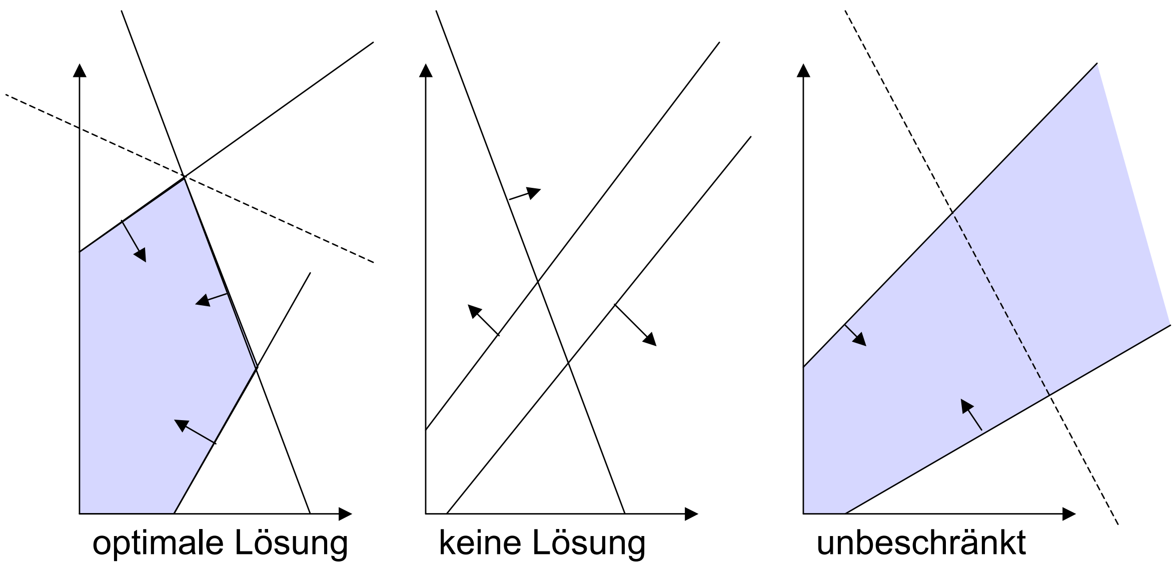


Graphisches Beispiel (4)



Lösungsmengen

Graphische Darstellung möglicher Lösungsmengen im 2-dimensionalen:



Graphische Interpretation

- Variablenbelegung $\mathbf{x}=(x_1,x_2,\ldots,x_n)$ entspricht Punkt im n-dimensionalen Raum
- Jede Nebenbedingung stellt einen Halbraum im n-dimensionalen Raum dar
- Ein Halbraum ist durch die Hyperebene beschränkt, die entsteht, wenn die Nebenbedingung durch Gleichheit erfüllt wird.

  (Hyperebenen sind (n-1)-dim. Teilmengen des n-dim.Raums)
- Die Schnittmenge aller Halbräume stellt den Bereich zulässiger Lösungen dar (**zulässiger Bereich**)
- Man nennt diesen Bereich auch **Simplex**

Konvexität des zulässigen Bereichs

Lemma:

Die Punktmenge, die durch ein Simplex S beschrieben wird, ist konvex. D.h.:

$$x, y \in S \Rightarrow \overline{xy} \in S$$

Beweis:

- Ein Simplex ist der Schnitt mehrerer Halbräume
- Halbräume sind konvex
- Schnitte konvexer Mengen A,B sind konvex:

$$a,b \in A \cap B \quad \Rightarrow \quad \overline{ab} \in A \quad \wedge \quad \overline{ab} \in B$$

$$\Rightarrow \overline{ab} \in A \cap B$$

Lokales Optimum = globales Optimum

Lemma:

Ein lokales Optimum ist gleichzeitig ein globales Optimum

Beweis: (Widerspruch)

Sei x eine Lösung mit lokalem Optimum und y eine Lösung mit globalem Optimum. Es sei außerdem $c^T x < c^T y$.

Betrachte $z \in \overline{xy}$ aus einer ε-Umgebung von x.

$$0 < \|x - z\| \leq \varepsilon$$

$$c^T z = c^T(\lambda x + (1 - \lambda y)) = \lambda c^T x + (1 - \lambda) c^T y$$

$$> \lambda c^T x + (1 - \lambda) c^T x = c^T x$$

z hat einen größeren Zielfunktionswert als x, also ist x kein lokales Optimum

Optimum ist Ecke des Simplex

Lemma:

Wenn es für ein lineares Programm eine optimale Lösung gibt, so gibt es eine Ecke des zugehörigen Simplexes, deren Zielfunktionswert optimal ist.

Beweis:

- Schließe innere Punkte als optimale Lösung aus
- Zeige, daß optimale Lösungsmenge mindestens eine Ecke enthält

Optimale Lösung liegt auf dem Rand

- Alle zulässigen Lösungen mit demselben Zielfunktionswert liegen auf einer Hyperebene H, die orthogonal zum Vektor c der Zielfunktion ist. Sei H_x eine solche Hyperebene, die die Lösung x enthält.
- Je weiter H vom Ursprung entfernt ist, desto größer ist der zugehörige Zielfunktionswert.
- Sei P ein Punkt im Inneren des Simplex. Dann teilt H_P eine ε-Umgebung von P in zwei Hälften. Alle Punkte innerhalb der Hälfte, die weiter vom Ursprung entfernt ist, besitzen einen größeren Zielfunktionswert.

Optimale Lösungsmenge enthält Ecke

- Eine optimale Lösung muß also auf dem Rand liegen.
- H_{opt} schneidet also nicht das Simplex, sondern berührt es nur.
- Die Schnittmenge von H_{opt} mit dem Simplex ist also eine Facette oder ein Teil davon (Ecke, Kante,…)
- Es muß aber eine Ecke enthalten sein, da jede Facette des Simplexes mindestens eine Ecke enthält.

Freie Universität Berlin

Simplexalgorithmus

- Klassisches Verfahren zur Lösung linearer Programme
- Laufzeit ist im schlechtesten Fall **nicht** polynomial
- Meist liefert der Simplexalgorithmus aber sehr schnell ein Ergebnis

Freie Universität Berlin

Idee des Simplexalgorithmus

- Eingabe: lineare Programm in Schlupfform
- Ausgabe: optimale Lösung
- Algorithmus beginnt in einer Ecke
- bewegt sich in jedem Schritt von einer Ecke in die nächste Ecke
- Dabei wird der Zielfunktionswert meist erhöht (auf jeden Fall nie verkleinert)
- Terminiert, wenn lokales Optimum gefunden

Grundidee der Iteration

- Jede Iteration besitzt eine *Basislösung*
- Diese erhält man, indem man jede Nichtbasisvariable auf null setzt und so die Werte der Basisvariablen aus den Nebenbedingungen berechnet
- Jede Basislösung entspricht einer Ecke des Simplex
- Algebraisch ist jede Iteration eine äquivalente Umformung einer Schlupfform in eine andere

Grundidee der Iteration (Fortsetzung)

- Nun vertauschen wir die Rollen einer Nichtbasisvariablen und einer Basisvariablen
- Damit der Zielfunktionswert bei jeder Iteration nicht kleiner ist als in der vorherigen, wählen wir die Nichtbasisvariable (nach der wir umformen) so aus, dass der Zielfunktionswert größer wird, wenn wir den Wert dieser Variable von null aus erhöhen.

Basislösung ist Ecke des Simplex

- Lineares Programm mit n Variablen (man befindet sich im n-dim. Raum)
- Basislösung ist Schnittpunkt von n Nebensbedingungshyperebenen
- Bei Basislösung haben mindestens n Variablen den Wert 0
  $\Rightarrow$ mind. n Ungleichungen mit Gleichheit erfüllt

Beispiel für den Simplexalgorithmus

Gegeben: lineares Programm in Schlupfform

$z = 3x_1 + x_2 + 2x_3$
$x_4 = 30 - x_1 - x_2 - 3x_3$
$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$
$x_6 = 36 - 4x_1 - x_2 - 2x_3$

Basisvariablen: x_4, x_5, x_6
Nichtbasisvariablen: x_1, x_2, x_3

- Es gibt unendlich viele Lösungen
- Lösung ist *zulässig*, wenn
  $x_1, x_2, x_3, x_4, x_5, x_6 >= 0$

Bestimmen der Basislösung

- setze alle Nichtbasisvariablen auf 0
- berechne Werte der Basisvariablen
- im Beispiel:
  die Basislösung ist
  $(x_1, x_2, x_3, x_4, x_5, x_6) = (0, 0, 0, 30, 24, 36)$

  Sie hat den Zielfunktionswert
  $z = 3*0 + 1*0 + 2*0 = 0$

Ist die Basislösung zulässig, bezeichnen wir sie als **zulässige Basislösung**

Iteration

- Ziel der Iteration: Basislösung hat einen größeren Zielfunktionswert
- Wir wählen Nichtbasisvariable x_e aus, deren Koeffizient in der Zielfunktion nicht negativ ist und erhöhen den Wert um so viel, dass man gerade noch keine Nebenbedingung verletzt
- Variable x_e wird zur Basisvariable
- Basisvariable x_a, die zur stärksten Nebenbedingung gehört, wird dafür zur Nichtbasisvariable

Beispiel-Iteration

Zurück zum Beispiel:

$z = 3x_1 + x_2 + 2x_3$ (1)
$x_4 = 30 - x_1 - x_2 - 3x_3$ (2)
$x_5 = 24 - 2x_1 - 2x_2 - 5x_3$ (3)
$x_6 = 36 - 4x_1 - x_2 - 2x_3$ (4)

Wir erhöhen x_1
Es muss gelten:
$x_1 \le 30$ wegen x_4
$x_1 \le 12$ wegen x_5
$x_1 \le 9$ wegen x_6

Die letzte Bedingung ist am stärksten.
Wir vertauschen die „Rollen" von x_1 und x_6

Beispiel-Iteration (Fortsetzung)

- Wir formen (4) um:

$$x_1 = 9 - x_2/4 - x_3/2 - x_6/4$$

- Wir substituieren bei allen anderen Gleichungen x_1 mit dieser Gleichung
- So erhalten wir unser Programm in folgender Form:

$$z = 27 + x_2/4 + x_3/2 - 3x_6/4$$
$$x_1 = 9 - x_2/4 - x_3/2 - x_6/4$$
$$x_4 = 21 - 3x_2/4 - 5x_3/2 + x_6/4$$
$$x_5 = 6 - 3x_2/2 - 4x_3 + x_6/2$$

Basisaustausch

- Diese Umformung bezeichnet man als **Pivotieren** oder **Basisaustausch**
- Der Basisaustausch wählte eine Nichtbasisvariable x_e (Eingangsvariable) und eine Basisvariable x_a (Ausgangsvariable) und vertauschte ihre Rollen

- Die Basislösung des „neuen“ Programms ist

  $(x_1, x_2, x_3, x_4, x_5, x_6) =(9, 0, 0, 21, 6, 0)$

  mit Zielfunktionswert $z = 27$

Iterationsende

- Nun erhöhen wir den Wert von x_2:

  Es muss gelten:

  $x_2 \leq 36$ wegen x_1

  $x_2 \leq 28$ wegen x_4

  $x_2 \leq 4$ wegen x_5

- Wir tauschen also die Rollen von x_2 und x_5

  $x_2 = 4 – 8x_3/3 – 2x_5/3 + x_6/3$

Iterationsende

$z = 28 - x_3/6 - x_5/6 - 2x_6/3$
$x_1 = 8 + x_3/6 + x_5/6 - x_6/3$
$x_2 = 4 - 8x_3/3 - 2x_5/3 + x_6/3$
$x_5 = 18 - x_3/2 + x_5/2$

- Die Basislösung dieses Programms ist
  $(x_1, x_2, x_3, x_4, x_5, x_6) = (8, 4, 0, 0, 18, 0)$
  mit Zielfunktionswert $z = 28$
- Alle Koeffizienten der Zielfunktion sind nun negativ
- Basislösung ist optimale Lösung (Beweis: Teil 2)

Bedeutung der Schlupfvariablen

- Man beachte: die Werte der Schlupfvariablen in der entgültigen Lösung geben den Spielraum der Ungleichungen an
- z.B. $x_4 = 18$

  In der ursprünglichen Form

  $x_1 + x_2 + 3x_3 \leq 30$

  $8 + 4 = 12 \quad \leq 30$

  Die Differenz zwischen den beiden Seiten ist genau 18

Hilfssätze zur Terminierung (1)

Hilfssatz:

I sei Menge von Indizes. Für alle Belegungen der x_i soll gelten: $$\sum_{i \in I} a_i x_i = c + \sum_{i \in I} b_i x_i$$

Dann folgt daraus: $a_i = b_i$ und $c = 0$

Beweis:

Seien z.B. alle $x_i = 0$ $\Rightarrow c = 0$

Seien alle $x_i = 0$ bis auf ein $x_j = 1$

Dann muß gelten: $a_j = b_j$

Da x_j beliebig gewählt werden kann, gilt dies für alle a_i, b_i

Hilfssätze zur Terminierung (2)

Hilfssatz:

Die Schlupfform eines linearen Programms ist durch die Menge der Basisvariablen eindeutig bestimmt.

Beweis: (Widerspruch)

Basisvariablen x_i und 2 verschiedene Schlupfformen: $x_i = b_i - \sum_{j \in N} a_{ij} x_j$ und $x_i = b_i' - \sum_{j \in N} a_{ij}' x_j$

Gleichsetzen: $$\sum_{j \in N} a_{ij}' x_j = b_i - b_i' + \sum_{j \in N} a_{ij} x_j$$

Nach vorigem Hilfssatz: $a_{ij} = a_{ij}'$ und $(b_i - b_i') = 0$

Terminierung des Algorithmus

Lemma:

Falls der Simplexalgorithmus nach maximal $\binom{n+m}{m}$ Iterationen nicht terminiert, dann kreiselt er.

Beweis:

B ist Menge der Basisvariablen . |B|=m. Es gibt n+m viele Variablen. Es gibt demnach

$\binom{n+m}{m}$ eindeutige Schlupfformen

Werden mehr Iterationen durchgeführt, kreiselt der Algorithmus also.

Andere Algorithmen

Außer dem Simplexalgorithmus gibt es noch andere Algorithmen zur Lösung linearer Programme

Diese laufen sogar im Worst-Case in polynomieller Zeit. In der Praxis erweist sich der Simplexalgorithmus jedoch als schneller, weil er eine erwartete lineare Zeit hat.

- Ellipsoidmethode (Khachian, 1979)
- Innerer-Punkt-Methode (Kharmarkar, 1984)

Literaturverzeichnis

- Cormen/Leiserson/Rivest/Stein: Algorithmen- Eine Einführung
- Skript: Einführung in die Lineare Programmierung, Lehrstuhl Informatik 1, Algorithmen&Komplexität, RWTH Aachen
- Skript: Lineare Programmierung,WS02/03, Winfried Hochstättler, BTU Cottbus
- Skript: Ausarbeitung zum Thema :Der Simplexalgorithmus als Teil des Seminars Strategiekonstruktion für unendliche Spiele, Sandip Sar-Dessai, RWTH Aachen
- Berthold Vöcking, Universität Dortmund, Crashkurs lineare Programmierung