Seminar Algorithmen - Lineare Programmierung Teil 2

Was wissen wir bereits?

Simplex-Algorithmus löst Probleme, die in lin. Gleichungssysteme überführt werden können.
Er iteriert über die Gleichungssysteme indem er äquivalente Umformungen von Schlupfformen durchführt.

Eingabe: Schlupfform eines lin. Programms
Ausgabe: optimale Lösung

Laufzeit: meist polynomial

Gesucht ist Lösung, die folgende lin. Funktion

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

unter den Nebenbedingungen

$$\sum_{j=1}^{n} a_{ij} x_j \leqslant b_i \qquad \text{und } i=1,2,...,m$$

$$x_j \geqslant 0 \qquad \text{für } j=1,2,...,n$$

maximiert.

Nach Überführung der Standardform des lin. Programms in die Schlupfform $s = b_i - \sum_{j=1}^{n} a_{ij} x_j$ kann

der Simplexalgorithmus nun die Nichtbasisvariablen maximieren.
Er nimmt mittels seiner Pivot-Funktion Basisvariablenvertauschungen vor und stellt dabei die Gleichungen um.
Sobald alle Zielfunktionskoeffizienten c_j <0 sind, liegt ein optimales Ergebnis vor.

Basisvariablenvertauschungen :
die Gleichung die die Maximierung einer Nichtbasisvar. am meisten einengt wird zur Vertauschung vorgenommen.
Die Substitututionen der neuen Basisvariablen folgt in allen Gleichungen.

Terminierungsformen des „Simplex“:

Simplexalgorithmus kreiselt nach $\binom{n+m}{m}$ Iterationen, wenn er unbeschränkt ist

Simplexalgorithmus terminiert mit zulässiger Lösung.

Was bleibt zu zeigen?

TODO: Der Umgang mit LP ohne zulässige Lösung

TODO: Der Umgang mit ungültiger initialer Basislösung

TODO: Liefert „Simplex“ immer optimales Ergebnis?

Motivation - lineare Programmierung

Der Simplexalgorithmus bietet sich als Lösung für viele reale Probleme an.
- z.B.: Flussprobleme,
- für die Erstellung von Flugplänen, USA 80er Jahre
- Kürzeste Pfade-Probleme
- div. Kostenkonstellationen in der Betriebswirtschaft

Sobald ein Problem einmal als lineares Programm mit polynomialer Größe formuliert ist, kann es mit „Simplex“ gelöst werden.

Besonders Interessant: Das Problem des Multiplen Warenflusses kann bisher nur von dem Simplexalgorithmus effiktiv gelöst werden.

Verallgemeinerung von Multiples Warenfluss-Problem :

Gerichteter Graph G = (V,E)
Jede Kante (u,v) e E besitzt Kapazität c(u,v) >= 0.
Bsp.: k Waren, Ware i mit (q_i, s_i, b_i)
q = Quelle, s = Senke, b = Bedarf

f(u,v) = $\sum_{i=1}^{k} f_i(u, v)$ (Aggregat-Fluss)

da in der Problemstellung keine Minimieren vorhanden => Zielfunktion minimiere 0.
Man kann nun die Nebenbedingungen des Problems wie folgt formulieren:

$$
\begin{aligned}
\sum_{i=1}^{k} f_i(u,v) &\le c(u,v) && \text{für alle } u,v \in V ,\\
f_i(u,v) &= -f_i(v,u) && \text{für alle } i = 1,2,\ldots,k \text{ und}\\
& && \text{für alle } u,v \in V ,\\
\sum_{v \in V} f_i(u,v) &= 0 && \text{für alle } i = 1,2,\ldots,k \text{ und}\\
& && \text{für alle } u \in V - \{s_i,t_i\} ,\\
\sum_{v \in V} f_i(s_i,v) &= d_i && \text{für alle } i = 1,2,\ldots,k ,\\
f_i(u,v) &\le c(u,v) && \text{für alle } u,v \in V \text{ und}\\
& && \text{für alle } i = 1,2,\ldots,k .
\end{aligned}
$$

Sobald also Simplex 0 liefert ist eine Lösung vorhanden, sonst nicht.

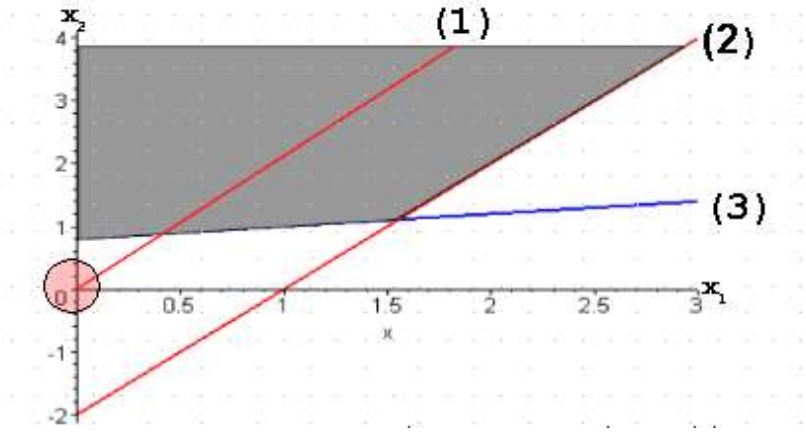
Simplex-Algorithmus – ungültige initiale Basislösung

initiale Basislösung ungültig - Beispiel

Standard-Form:
(1) maximiere $2x_1 - x_2$

(2) $2x_1 - x_2 <= 2$
(3) $x_1 - 5x_2 <= -4$
$x_1, x_2 >= 0$

Schlupfform:
(1) $z = 2x_1 - x_2$
(2) $x_3 = 2 - 2x_1 + x_2$
(3) $x_4 = -4 - x_1 + 5x_2$
$x_1,x_2,x_3,x_4 >= 0$
Basislösung: {0,0,-4!,2}



Simplex-Algorithmus – unlösbar

Der Simplex-Algorithmus liefert „unlösbar“ zurück wenn:
- Nebenbedingungen im „Widerspruch“ stehen
- der Lösungsbereich Unbeschränkt ist
Wir benötigen also eine Initialisierungsprozedur für jedes Lineare Programm L, wir nennen es „INITIALIZE-SIMPLEX“ (nach Cormen).

Was leistet INITIALIZE-SIMPLEX?
-Die Schlupfform wird an die Testprozedur übergeben
ist die Basislösung zulässig: Rückgabe der original Schlupfform
ist die Basislösung nicht zulässig: weitergeben der Schlupfform an eine neue Hilfsfunktion
wird nach der Hilfsfunktion die Originale Schlupfform zurückgegeben ist das Programm lösbar, sonst nicht.
Die „Hilfsfunktion“:
ein Lineares Programm, dass prüft, ob L lösbar ist.
Innerhalb der Hilfsfunktion wird L verändert und daher L_h genannt.
L_h hat zusätzlich zu den n Variablen von L, die Hilfsvariable x_0.
maximiere $-x_0$ (b : Basisvariablen, a : Koeffizienten, x : Nicht-Basisvariblen)

$$\sum_{j=1}^{n} a_{ij} x_j - x_0 \leqslant b_i \quad \text{für i = 1, 2, ..., m}$$

$$x_j \geqslant 0 \quad \text{für j = 0, 1, ..., n}$$

Wenn der optimale Zielfunktionswert des Hilfsprogramms negativ ist, dann hat das ursprüngliche lineare Programm keine zulässige Lösung. Ist die Lösung gleich 0, dann hat das ursprüngliche LP eine Lösung.

Beweis:
wenn es eine Lösung von LP gibt, dann muss es sie auch geben, wenn $-x_0$ 0 wird. Da man $-x_0$ maximiert und die Nebenbedingung u.a. x_0 >= 0 lautet, ist die optimale Lösung der Zielfunktion bestenfalls 0. Gibt es also eine Lösung, muss nach Anwendungen des Simplex-A. (Schlupfumformung, Pivotieren etc.) $x_0 = 0$ sein.

Simplex-Algorithmus – unbeschränkt

Ein lineares Programm LP ist unbeschränkt, wenn die Nebenbedingungen den Umfang nicht ausreichend einschränken und „Simplex“ die Eingangsvariablen beliebig erhöhen kann.

Der Fundamentalsatz der linearen Progammierung

Der Fundamentalsatz der linearen Programmierung besagt über ein lineares Programm LP
1. LP besitzt entweder eine optimale Lösung (endlicher Zielfunktionswert) oder
2. LP ist unlösbar,
3. LP ist unbeschränkt.

Das Prinzip der Dualität

Allgemein gilt:

• Die rechten Seiten der Ungleichung der primalen Aufgabe sind die Zielfunktionskoeffizienten der dualen Aufgabe. Umgekehrt sind die Zielfunktionskoeffizienten der primalen Aufgabe die rechten Seiten der Ungleichung der dualen Aufgabe.

• Die Koeffizienten der linken Seiten der Ungleichung der primalen Aufgabe ergeben spaltenweise die Koeffizienten der linken Seiten der Ungleichung der dualen Aufgabe.
Daraus ergibt sich auch, dass die Anzahl der Strukturvariablen der primalen Aufgabe gleich der Anzahl der Schlupfvariablen der dualen Aufgabe ist, und umgekehrt.

Formal bedeutet das:

$$(1) \quad \sum_{j=1}^{n} c_j x_j$$

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

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

$$(3) \quad \sum_{i=1}^{m} b_i y_i$$

$$(4) \quad \sum_{i=1}^{m} a_{ij} y_i \geq c_j \qquad j = 1, 2, \cdots, n$$

$$y_i \geq 0 \qquad i = 1, 2, \cdots, m$$

Als schwache Dualität wird das Prinzip bezeichnet, wonach das Minimierungs-Programm eines Problems (duales Programm) die obere Schranke des Maximierungs-Programms eines Problems (primales Programm) darstellt:

$$\sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{m} b_i y_i$$

(nach 4)
$$\sum_{j=1}^{n} c_j x_j \leq \sum_{j=1}^{n} (\sum_{i=1}^{m} a_{ij} y_j) x_j$$

$$= \sum_{i=1}^{m} (\sum_{j=1}^{n} a_{ij} x_j) y_j$$

(nach 2)
$$\leq \sum_{i=1}^{m} b_i y_i$$

Als Weiterführung dieses Prinzips kann man sich überlegen, dass wenn beide Programme den gleichen Zielfunktionswert erreichen, dieser optimal sein muss, also

$$\sum_{j=1}^{n} c_j x_j = v' + \sum_{j=1}^{n+m} c'_j x_j$$

Letztendlich kann man u.A. hieraus das Prinzip der starken Dualität, bzw. Dualität der lineare Programmierung folgern:

$$\sum_{j=1}^{n} c_j \bar{x}_j = \sum_{i=1}^{m} b_i \bar{y}_i$$

Ist x die optimale Lösung des primalen linearen Programms und y die optimale Lösung des dualen, dann haben beide immer den gleichen Zielfunktionswert. Daraus folgt der Beweis des 1. Satz des Fundamentaltheorems der linearen Programmierung.

Quellen:

Cormen, Thomas; Leiserson, Charles;Rivest, Ronald; Stein, Clifford (2001): Introduction to Algorithms, Second Edition.
Schlipf, Lena; Jankovic, Benjamin (2005): Lineare Programmierung Teil 1, http://page.mi.fu-berlin.de/~alt/vorlesungen/sem05/folien3.pdf
Jiangsheng, Yu (2003): Linear Programming. http://icl.pku.edu.cn/member/yujs/papers/pdf/ComAlg9.pdf
Riegler, Peter (2001): Dualität. Ergänzungen zum Kurs Operations Research, **http://people.freenet.de/riegler/dualtext.pdf**
Wisniewski, Timothy (1996): The Simplex Java Applet, **http://www-fp.mcs.anl.gov/otc/Guide/CaseStudies/simplex/applet/SimplexTool.html**