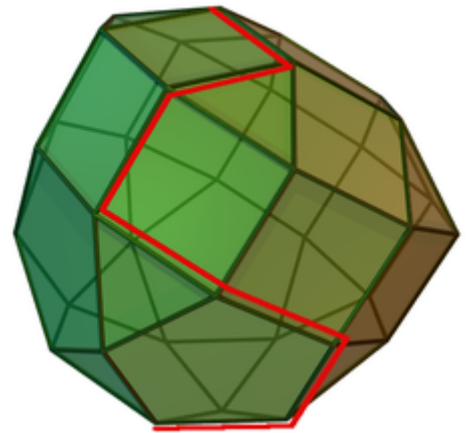


# Simplex-Verfahren

Ein **Simplex-Verfahren** (auch **Simplex-Algorithmus**) ist ein Optimierungsverfahren der Numerik zur Lösung linearer Optimierungsprobleme, auch als Lineare Programme (LP) bezeichnet. Es löst ein solches Problem nach endlich vielen Schritten exakt oder stellt dessen Unlösbarkeit oder Unbeschränktheit fest. Die Grundidee der Simplex-Verfahren wurde 1947 von George Dantzig vorgestellt; seitdem haben sie sich durch zahlreiche Verbesserungen zu den wichtigsten Lösungsverfahren der linearen Optimierung in der Praxis entwickelt. Simplex-Verfahren sind Pivotverfahren.

Obwohl bisher für jede Variante des Verfahrens ein Beispiel konstruiert werden konnte, bei dem der Algorithmus exponentielle Laufzeit benötigt, läuft ein Simplex-Algorithmus in der Praxis meist schneller als andere Verfahren, obwohl es zur Lösung einzelner linearer Programme auch andere konkurrenzfähige Methoden gibt, wie z. B. Innere-Punkte-Verfahren. Der große Vorteil eines Simplex-Algorithmus liegt jedoch darin, dass er bei leichter Veränderung des Problems – beispielsweise dem Hinzufügen einer zusätzlichen Bedingung – einen „Warmstart“ von der letzten verwendeten Lösung durchführen kann und daher meist nur wenige Iterationen zur erneuten Lösung benötigt, während andere Verfahren von vorne beginnen müssen. Darüber hinaus nutzt ein Simplex-Verfahren die engen Zusammenhänge zwischen einer linearen Optimierungsaufgabe und seiner dualen Aufgabe aus und löst grundsätzlich beide Probleme gleichzeitig. Beide Eigenschaften sind in der ganzzahligen linearen oder auch nichtlinearen Optimierung dort von Bedeutung, wo sehr viele ähnliche lineare Aufgaben in Folge gelöst werden müssen.

Die geometrische Grundidee des Algorithmus besteht darin, von einer beliebigen Ecke eines Polyeders, das durch die lineare Optimierungsaufgabe definiert wird, entlang seiner Kanten zu einer optimalen Ecke zu laufen. Der Name des Verfahrens rührt daher, dass die nichtnegativen Linearkombinationen der Basisspalten in jeder Iteration einen simplicialen Kegel beschreiben. Ein Namensvetter dieses Verfahrens namens Downhill-Simplex-Verfahren (Nelder und Mead 1965<sup>[1]</sup>) basiert ebenfalls auf einem Simplex, ist aber ein iteratives Verfahren zur nichtlinearen Optimierung.



Das Primalsimplex-Verfahren läuft von einer Ecke eines LP-Polyeders zur nächsten, bis keine Verbesserung mehr möglich ist.

## Inhaltsverzeichnis

### Geschichte

### Grundidee

### Laufzeit

### Mathematische Beschreibung

- Bestimmung einer Startlösung (Phase I)
- Bestimmung einer Optimallösung (Phase II)
  - Basen, Basislösungen und Ecken
  - Iterative Simplexschritte
  - Simplextableau
  - Ein einzelner Simplexschritt

**Beispielrechnung**

- Überführung in Gleichungen
- Bestimmung einer zulässigen Ausgangslösung (Phase I)
- Verbesserung der Ausgangslösung (Phase II)
  - Auswahl einer Pivotspalte und -zeile
  - Durchführung eines Austauschschrittes
  - Wiederholung der Simplexschritte
- Einträge in Bruchzahlform
- Beispielhafte Lösung in Java

**Duale Information im Tableau****Varianten und Verbesserungen**

- Auswahl des Pivotelementes
  - Spaltenauswahl
  - Zeilenauswahl
- Variablenschranken
- Duales Simplex-Verfahren
- Revidiertes Simplex-Verfahren
- LR-Zerlegungen
- Preprocessing

**Literatur****Weblinks****Einzelnachweise**

## Geschichte

---

Die Grundlagen der linearen Optimierung wurden 1939 von dem russischen Mathematiker Leonid Witaljewitsch Kantorowitsch in seinem Buch „Mathematische Methoden in der Organisation und Planung der Produktion“ gelegt. Kurz danach (1941) präsentierte der US-Amerikaner Frank L. Hitchcock (1875–1957) eine Arbeit zu einem Transportproblem. Im Jahre 1947 veröffentlichte George Dantzig das Simplex-Verfahren, mit dem lineare Programme erstmals systematisch gelöst werden konnten. Eine der ersten dokumentierten Anwendungen der neuen Methode war das *Diäten-Problem* von George Stigler, dessen Ziel eine möglichst kostengünstige Nahrungszusammensetzung für Soldaten war, die bestimmte Mindest- und Höchstmengen an Vitaminen und anderen Inhaltsstoffen erfüllte. An der optimalen Lösung dieses linearen Programms mit neun Ungleichungen und 77 Variablen waren damals neun Personen beschäftigt, die zusammen etwa 120 Manntage Rechenarbeit benötigten.<sup>[2]</sup> Interesse an dieser Arbeit zeigte zunächst das amerikanische Militär, speziell die US Air Force, die militärische Einsätze optimieren wollte. In den Folgejahren entwickelten John von Neumann und Oskar Morgenstern das Verfahren weiter.

Mit dem Aufkommen von Computern Mitte der 1950er Jahre konnten auch größere Probleme gelöst werden. Es wurden spezielle Varianten der Simplexmethode entwickelt, wie das revidierte Simplex-Verfahren, das sehr sparsam mit dem damals knappen und teuren Hauptspeicher umging. Im Jahre 1954 brachte William Orchard-Hays die erste kommerzielle Implementierung dieses Verfahrens auf den Markt. Im selben Jahr veröffentlichten Carlton Lemke und E. M. L. Beale das duale Simplex-Verfahren, das sich heute – nach weiteren Verbesserungen – zu einer der Standardmethoden zur Lösung linearer Programme entwickelt hat.

Die theoretische Komplexität des Simplex-Verfahrens war lange Zeit unklar. Erst im Jahre 1972 konstruierten Victor Klee und George Minty ein Beispiel, bei dem der Algorithmus alle exponentiell vielen Ecken eines Polyeders abläuft, und zeigten damit die exponentielle Laufzeit des Verfahrens. Ähnliche Beispiele wurden bisher für alle bekannten Varianten des Verfahrens gefunden.

Ab den 1970er Jahren profitierte der Simplex-Algorithmus – wie auch andere Verfahren der Linearen Optimierung – von algorithmischen Fortschritten der numerischen linearen Algebra, insbesondere bei der Lösung großer linearer Gleichungssysteme. Vor allem die Entwicklung numerisch stabiler LR-Zerlegungen für dünnbesetzte Matrizen trug maßgeblich zum Erfolg und der Verbreitung des Simplex-Verfahrens bei.

Seit Mitte der 1970er bis Anfang der 1990er Jahre wurde das Verfahren durch die Entwicklung neuer Pivotstrategien deutlich verbessert. Vor allem die wachsende Bedeutung der ganzzahligen linearen Optimierung in den 1980er Jahren sowie die Entwicklung des *dual steepest edge pricing* in der Implementierung von J. J. Forrest und Donald Goldfarb (1992) machten das duale Simplex-Verfahren zum ernsthaften Konkurrenten für andere Lösungsmethoden. Umgekehrt hatte diese Verbesserung des dualen Simplex-Algorithmus einen maßgeblichen Anteil am Erfolg von Schnittebenenverfahren und Branch-and-Cut zur Lösung ganzzahliger linearer Programme. Darüber hinaus sorgten neue Preprocessing-Methoden in den 1990er Jahren dafür, dass immer größere LPs gelöst werden konnten. Unter der – in praktischen Anwendungen fast immer erfüllten – Voraussetzung, dass die auftretenden LP-Matrizen dünnbesetzt sind, können heute lineare Programme mit mehreren hunderttausend Variablen oder Ungleichungen innerhalb weniger Stunden optimal gelöst werden.

## Grundidee

Die Simplex-Verfahren dienen zur Lösung linearer Optimierungsaufgaben, das ist die Suche nach reellen Variablenwerten, die ein System linearer Ungleichungen und Gleichungen erfüllen und dabei eine lineare Zielfunktion maximieren oder minimieren. Ausgegangen wird dabei von der Form

$$(LP) \max_{x \in \mathbb{R}^n} \{c^T x \mid Ax \leq b, x \geq 0\},$$

wobei  $A \in \mathbb{R}^{m \times n}$  eine Matrix mit reellen Einträgen ist,  $c \in \mathbb{R}^n$  der sogenannte Zielfunktionsvektor, und  $b \in \mathbb{R}^m$ ,  $b \geq 0$  ein Spaltenvektor mit *nichtnegativen Einträgen*  $b_i \geq 0$ . Ziel ist es, einen Punkt  $x$  zu finden, der das lineare Ungleichungssystem erfüllt und einen möglichst hohen Zielfunktionswert  $F(x) = c^T x$  hat.

Jedes lineare Programm kann durch einfache Umformungen in die Form

$$(LP) \max_{x \in \mathbb{R}^n} \{c^T x \mid Ax \leq b, x \geq 0\}$$

gebracht werden, bei welcher die Vorzeichen in  $b \in \mathbb{R}^m$  freilich *immer noch beliebig* sind. Das geschieht wie folgt:

- Ein Minimierungsproblem kann durch Multiplikation der Zielfunktion mit  $(-1)$  in ein Maximierungsproblem verwandelt werden.
- Ungleichungen  $\sum_j A_{ij}x_j \geq b_i$  können als  $\sum_j (-A_{ij})x_j \leq -b_i$  geschrieben werden.
- Vorhandene Gleichungsgruppen  $\forall i \sum_j A_{ij}x_j = b_i$  können in Ungleichungsgruppen  $\forall i \sum_j A_{ij}x_j \leq b_i$  mit  $\sum_j (-\sum_i A_{ij})x_j \leq -b_i$  aufgespalten werden.
- Variablengruppen  $x_j$  mit beliebigem Wertebereich können über eine zusätzliche Variable und  $x_j = u_j - w$  durch nichtnegativen Variablen  $\forall j \ u_j \geq 0, w \geq 0$  ersetzt werden.

In Gegensatz zu diesen Umformungen ist die *immer vorausgesetzte* Startbedingung  $b \geq 0$  nur über die Lösung einer Hilfsaufgabe in einer sogenannten "Phase 1" zu erreichen; dabei ist diese Hilfsaufgabe grundsätzlich ebenso aufwändig wie die eigentliche Optimierung.

Geometrisch lässt sich die Menge der zulässigen Lösungen, also die Menge der Punkte mit nichtnegativen Koordinaten, die das lineare Ungleichungssystem  $Ax \leq b$  erfüllen, als konvexes Polyeder (mehrdimensionales Vieleck)  $P$  interpretieren, dessen Dimension durch die Anzahl der Variablen nach oben begrenzt ist. Ziel ist es, einen

Punkt  $x$  in  $P$  mit möglichst hohem Zielfunktionswert  $F(x) = c^T x$  zu finden. Anschaulich entspricht dies der Verschiebung der Hyperebene  $c^T x = 0$  so weit wie möglich in Richtung des Vektors  $c$ , so dass sie gerade noch das Polyeder berührt. Alle Berührungspunkte sind dann optimal.

Für die Menge der zulässigen Lösungen gibt es drei Möglichkeiten:

1. das LP besitzt keine zulässigen Lösungen, d. h., das Polyeder ist leer (z. B.  $\max\{x \mid x \leq 1, x \geq 2\}$ ).
2. das LP ist unbeschränkt, d. h., es gibt Lösungen mit beliebig hohem Zielfunktionswert (z. B.  $\max\{x \mid x \geq 0\}$ ).
3. es gibt genau eine oder unendlich viele Optimallösungen, die dann alle auf einer gemeinsamen Seitenfläche (Ecke, Kante, ...) des Polyeders  $P$  liegen.

Man kann zeigen, dass es immer eine optimale Ecke gibt, falls das Optimierungsproblem überhaupt zulässige Lösungen besitzt und beschränkt ist. Man kann sich also bei der Suche nach Optimallösungen auf die Ecken des Polyeders beschränken, von denen es allerdings sehr viele geben kann.

Die anschauliche Grundidee des Simplex-Verfahrens besteht nun darin, sich schrittweise von einer Ecke des Polyeders zu einer benachbarten Ecke mit besserem Zielfunktionswert zu hangeln, bis es keinen besseren Nachbarn mehr gibt. Da es sich bei der linearen Optimierung um ein konvexes Optimierungsproblem handelt, ist die so gefundene lokal optimale Ecke dann auch global optimal, d. h., es gibt in ganz  $P$  keine andere Ecke mit besserem Zielfunktionswert mehr.

## Laufzeit

Die Zahl der Ecken eines Polyeders kann exponentiell in der Anzahl der Variablen und Ungleichungen sein. Beispielsweise lässt sich der  $n$ -dimensionale Einheitswürfel durch  $2n$  lineare Ungleichungen beschreiben, besitzt aber  $2^n$  Ecken. Klee und Minty konstruierten im Jahre 1972 einen verzerrten Einheitswürfel, den sogenannten *Klee-Minty-Würfel*, bei dem die von Dantzig vorgestellte Variante des Simplex-Verfahrens tatsächlich alle diese Ecken besuchte.<sup>[3]</sup> Ähnliche Beispiele wurden bisher für alle Zeilen- und Spaltenauswahlregeln gefunden. Dies bedeutet, dass der Simplex-Algorithmus in allen bisher bekannten Varianten im schlechtesten Fall exponentielle Laufzeit besitzt.

Bei degenerierten linearen Programmen, wie sie in der Praxis häufig auftreten, kann es zu sogenannten *Zyklen* kommen, bei dem das Simplex-Verfahren immer wieder dieselbe Ecke betrachtet und dadurch nicht terminiert. Dies lässt sich aber durch Anwendung der lexikographischen Zeilenauswahlregel oder durch absichtliche numerische Störungen verhindern.

Aus theoretischer Sicht ist das Simplex-Verfahren daher beispielsweise den Innere-Punkte-Verfahren mit polynomialer Laufzeit unterlegen. Aus praktischer Sicht hat es sich aber in vielen Fällen als schneller erwiesen. Der größte Vorteil des Simplex-Algorithmus gegenüber anderen Verfahren liegt jedoch darin, dass es bei kleinen Änderungen der Eingabedaten im Laufe des Algorithmus einen „Warmstart“ erlaubt, also die letzte berechnete Basis als Ausgangspunkt für wenige weitere (primale oder duale) Iterationen nehmen kann, während beispielsweise Innere-Punkte-Verfahren in solch einem Fall von vorne anfangen müssen. Dieser Fall tritt sehr häufig auf, wenn sehr viele ähnliche lineare Programme in Folge gelöst werden müssen, beispielsweise im Rahmen von Schnittebenenverfahren, Branch-and-Bound oder Branch-and-Cut.

In der Praxis hängt die Laufzeit des Simplex-Verfahren oft im Wesentlichen linear von der Anzahl der Zeilen ab. Tatsächlich zeigten Borgwardt und andere in den 1980er Jahren, dass solche Fälle wie der Klee-Minty-Würfel extrem selten sind und dass einige Varianten des Simplex-Algorithmus unter bestimmten Annahmen an den Input im Mittel nur polynomiale Laufzeit benötigen. Es ist aber bis heute unklar, ob es eine Variante mit polynomialer Laufzeit für alle Instanzen gibt.

## Mathematische Beschreibung

Das Simplex-Verfahren setzt sich aus zwei Phasen zusammen:

- Phase I bestimmt eine zulässige Startlösung oder stellt fest, dass das Problem keine Lösung besitzt,
- Phase II verbessert eine bestehende Lösung immer weiter, bis keine Verbesserung der Zielfunktion mehr möglich ist oder die Unbeschränktheit des Problems festgestellt wird.

Die Big-M-Methode bietet eine Möglichkeit, beide Phasen zu verbinden, sprich den Simplex anzuwenden, auch wenn zunächst keine zulässige Startlösung gegeben ist.

## Bestimmung einer Startlösung (Phase I)

Zunächst muss eine zulässige Startlösung berechnet werden, bevor man die Phase II starten kann. Eine Möglichkeit dafür ist, für jede Zeile  $i$  eine künstliche Variable  $z_i$  einzuführen und dann folgendes lineare Programm zu betrachten:

$$(LP1) \min \left\{ \sum_{i=1}^m z_i \mid Ax + z = b, x, z_1, \dots, z_m \geq 0 \right\}.$$

In einer Optimallösung dieses Hilfsproblems sind die künstlichen Variablen so klein wie möglich, wobei sie immer nichtnegativ bleiben müssen. Das ursprüngliche Problem (LP) besitzt genau dann eine zulässige Lösung, wenn es eine Lösung des Hilfsproblems mit  $z = 0$  gibt, bei der also keine künstlichen Variablen verwendet werden.

Das Hilfsproblem (LP1) besitzt für  $b \geq 0$  eine einfache zulässige Startlösung, nämlich  $(x, z) = (0, b)$ . Darüber hinaus gilt für jede zulässige Lösung des Hilfsproblems  $z \leq b$ , so dass die Zielfunktion nach oben durch  $\sum_{i=1}^m b_i$  beschränkt ist. Dieses lineare Programm besitzt also eine Optimallösung, die ausgehend von der Startlösung  $(0, b)$  mit Phase II des Hilfsproblems bestimmt werden kann. Findet man dabei eine Optimallösung  $(x^*, z^*)$  mit  $z^* = 0$ , ist  $x^*$  offensichtlich eine zulässige Startlösung für das Ausgangsproblem (LP), mit der dessen Phase II gestartet werden kann. Gelingt dies nicht, so ist das Ausgangsproblem nicht lösbar und man kann aufhören.

## Bestimmung einer Optimallösung (Phase II)

Phase II ist ein iteratives Verfahren, das in jedem Schritt versucht, aus einer zulässigen Lösung eine neue Lösung mit besserem Zielfunktionswert zu konstruieren, bis dies nicht mehr möglich ist. Dies entspricht im Wesentlichen der wiederholten Lösung eines linearen Gleichungssystems mit Hilfe des Gaußschen Eliminationsverfahrens. Dabei kann auch evtl. die Unbeschränktheit des Optimierungsproblems festgestellt werden. Zur Erklärung dieser Phase ist die Definition einiger Begriffe notwendig.

### Basen, Basislösungen und Ecken

→ Hauptartikel: Zulässige Basislösung

In der Phase II des Simplex-Verfahrens spielt der Begriff der *Basis* eine wesentliche Rolle. Eine Basis  $B$  von  $A$  ist eine Teilmenge der Spalten von  $A$ , die eine reguläre Untermatrix  $A_B$  bilden.  $B$  wird dabei als Indexvektor über die Spalten von  $A$  dargestellt. Die Variablen, die zu den Basisspalten gehören, also in  $B$  enthalten sind, heißen *Basisvariablen*, die anderen *Nichtbasisvariablen*. Die *Basislösung* zu einer gegebenen Basis  $B$  ist der eindeutige Vektor  $x$ , dessen Basisvariablen sich als  $A_B^{-1}b$  ergeben und dessen Nichtbasisvariablen alle den Wert 0 haben (also  $A_B \cdot x_B = b$  und  $x_N = 0$ ). Solch eine Basislösung ist also eine zulässige Lösung des Gleichungssystems  $Ax = b$  mit höchstens  $m$  Nicht-Null-Einträgen. In dem Fall, dass alle Komponenten von  $x_B$  nichtnegativ sind, ist  $x \geq 0$  auch eine zulässige Lösung von (LP), also ein Punkt des Polyeders  $P$ .

Man kann zeigen, dass jede zulässige Basislösung einer Ecke (Extremalpunkt) des Polyeders entspricht und umgekehrt. Eine Basislösung heißt *nicht degeneriert* (*nicht entartet*), wenn sie genau  $m$  Nicht-Null-Basiseinträge besitzt, also  $x_B > 0$  gilt. In diesem Fall gibt es eine eindeutige zugehörige Basis. Sind alle Basislösungen nicht degeneriert, gibt es also eine bijektive Abbildung zwischen Basen der Matrix  $A$  und Ecken des Polyeders  $P$ .

Ist eine Basislösung dagegen *degeneriert*, so hat sie weniger als  $m$  Nicht-Null-Einträge und kann zu mehreren Basen gehören. In diesem Fall definiert also jede Basis der Matrix  $\mathbf{A}$  genau eine Ecke des Polyeders  $\mathbf{P}$ , aber nicht umgekehrt. Dieser Fall tritt auf, wenn eine Ecke von mehr als  $n$  Ungleichungen definiert wird, was bei praktischen Planungsproblemen so gut wie immer der Fall ist.

### Iterative Simplexschritte

Die Phase II versucht iterativ in jedem *Austauschschritt*, aus einer bestehenden Basislösung, wie sie z. B. in Phase I bestimmt wurde, eine neue Basislösung mit mindestens genauso gutem Zielfunktionswert zu konstruieren, indem sie eine Basisvariable aus der Basis entfernt und dafür eine bisherige Nichtbasisvariable in die Basis aufnimmt. Dies wird so lange wiederholt, bis kein verbessernder Austausch mehr möglich ist.

In dieser Phase gibt es zu Beginn jeder Iteration ein sogenanntes *Simplextableau* zur aktuellen Basis  $\mathbf{B}$ , in dem die Berechnungen durchgeführt werden. Es entspricht im Wesentlichen dem linearen Gleichungssystem  $[\mathbf{A} \ \mathbf{I}] \mathbf{x} = \mathbf{b}$ ,  $[\mathbf{c}^T \ 0] \mathbf{x} = z$ , nach einer Basistransformation in die aktuelle Basis.

### Simplextableau

Für die Formulierung des Simplextableaus gibt es unterschiedliche Möglichkeiten; die hier vorgestellte Version basiert auf einem Vorlesungsskript<sup>[4]</sup> von Martin Grötschel. Jeder Simplexschritt geht von einer zulässigen Basis  $\mathbf{B}$  aus. Zu Beginn des Schrittes hat das zugehörige Simplextableau die folgende Form:

$$\left[ \begin{array}{ccc|c} \bar{\mathbf{A}} & \mathbf{I} & \mathbf{0} & \bar{\mathbf{b}} \\ \bar{\mathbf{c}} & \mathbf{0} & \mathbf{1} & \bar{\mathbf{f}} \end{array} \right]$$

Hierbei sind zur Vereinfachung der Darstellung die Spalten so umsortiert worden, dass alle Nichtbasisspalten links stehen und alle Basisspalten rechts.  $\mathbf{I}$  ist die  $(m \times m)$ -dimensionale Einheitsmatrix. Die  $(m \times n)$ -dimensionale Matrix  $\bar{\mathbf{A}}$  und die restlichen obigen Einträge sind definiert durch

$$\begin{aligned} \bar{\mathbf{A}} &= \mathbf{A}_B^{-1} \mathbf{A}_N \\ \bar{\mathbf{b}} &= \mathbf{A}_B^{-1} \mathbf{b} \\ \bar{\mathbf{c}} &= \mathbf{c}_N^T - \mathbf{c}_B^T \mathbf{A}_B^{-1} \mathbf{A}_N \\ \bar{\mathbf{f}} &= \mathbf{c}_B^T \mathbf{A}_B^{-1} \mathbf{b} \end{aligned}$$

Dabei ist

- $\mathbf{A}_B$  die reguläre Untermatrix von  $\mathbf{A}$ , die aus den Spalten der aktuellen Basis  $\mathbf{B}$  besteht,
- $\mathbf{A}_N$  die (meistens nicht quadratische) Untermatrix von  $\mathbf{A}$ , die aus den Nichtbasisspalten besteht,
- $\mathbf{c}_B$  die Teile des Zielfunktionsvektors  $\mathbf{c}$ , die aus den Basisspalten bestehen, und
- $\mathbf{c}_N$  die Teile des Zielfunktionsvektors  $\mathbf{c}$ , die aus den Nichtbasisspalten bestehen.

Die rechte Seite  $\bar{\mathbf{b}}$  enthält die Werte der Basisvariablen von der zu  $\mathbf{B}$  gehörenden Basislösung;  $\bar{\mathbf{f}}$  ist der Zielfunktionswert dieser Basislösung. Zu Beginn der Phase I bilden die Variablen  $\mathbf{z}_i$  eine zulässige Basis mit der Einheitsmatrix als Basismatrix und der zugehörigen Basislösung  $(\mathbf{x}, \mathbf{z}) = (\mathbf{0}, \mathbf{b})$ . Daher steht am Anfang auf der rechten Seite einfach der Vektor  $\mathbf{b}$ .

Die Einträge des Vektors  $\bar{\mathbf{c}}$  heißen *reduzierte Kosten* der Nichtbasisvariablen. Der Wert  $\bar{c}_j$  gibt die Verbesserung der Zielfunktion an, wenn Variable  $\mathbf{x}_j$  um eine Einheit erhöht wird. Sind zu Beginn eines Simplexschrittes alle reduzierten Kostenkoeffizienten negativ oder 0, sind daher die aktuelle Basis und die zugehörige Basislösung optimal, da die Aufnahme einer bisherigen Nichtbasisvariable in die Basis den Zielfunktionswert verschlechtern würde. Gibt es

dagegen Nichtbasisvariablen mit positiven reduzierten Kosten, wird im nächsten Simplexschritt eine von ihnen in die Basis aufgenommen und dafür eine andere Variable aus der Basis entfernt. Wenn die neue Variable innerhalb der Nebenbedingungen  $\mathbf{Ax} = \mathbf{b}$  erhöht werden kann, verbessert sich der Zielfunktionswert.

### Ein einzelner Simplexschritt

Für den eigentlichen Simplexschritt wählt man nun eine Spalte  $s$  der einzufügenden Nichtbasisvariable und eine Zeile  $r$  der aus der Basis zu entfernenden Basisvariablen. Seien  $\bar{a}_{ij}$  die (Matrix-) Elemente des aktuellen Simplextableaus. Dann heißt  $\bar{a}_{rs}$  das *Pivotelement* des Simplextableaus. Die Spalte  $s$  heißt *Pivotspalte*, die Zeile  $r$  heißt *Pivotzeile*. Ein Austauschschritt entspricht exakt einem Schritt beim Lösen eines linearen Gleichungssystems, bei dem man die Pivotzeile  $r$  nach der Variablen  $x_s$  auflöst und dann  $x_s$  in die restlichen Gleichungen einsetzt. Bei einem Austauschschritt berechnet sich das neue Simplextableau folgendermaßen:

Pivotelement:

$$(1) \quad \bar{a}_{rs} := \frac{1}{\bar{a}_{rs}}$$

Pivotzeile für  $j$  ungleich  $s$ :

$$(2) \quad \bar{a}_{rj} := \frac{\bar{a}_{rj}}{\bar{a}_{rs}}$$

$$(3) \quad \bar{b}_r := \frac{\bar{b}_r}{\bar{a}_{rs}}$$

Pivotspalte für  $i$  ungleich  $r$ :

$$(4a) \quad \bar{a}_{is} := -\frac{\bar{a}_{is}}{\bar{a}_{rs}}$$

$$(4b) \quad \bar{c}_s := -\frac{\bar{c}_s}{\bar{a}_{rs}}$$

Restliche Elemente der Matrix und reduzierte Kosten:

$$(5a) \quad \bar{a}_{ij} := \bar{a}_{ij} - \frac{\bar{a}_{is} \bar{a}_{rj}}{\bar{a}_{rs}}$$

$$(5b) \quad \bar{c}_j := \bar{c}_j - \frac{\bar{c}_s \bar{a}_{rj}}{\bar{a}_{rs}}$$

Rechte Seite:

$$(6) \quad \bar{b}_i := \bar{b}_i - \frac{\bar{b}_r \bar{a}_{is}}{\bar{a}_{rs}}$$

Diese Rechenschritte entsprechen der Anwendung des Gaußschen Eliminationsverfahrens, um die Pivotspalte  $s$  im Tableau in einen Einheitsvektor zu transformieren. Dadurch wird die inverse Matrix  $\mathbf{A}_B^{-1}$  zur neuen Basis  $\mathbf{B}'$  nicht komplett neu berechnet, sondern durch Modifikation der alten Basisinversen  $\mathbf{A}_B^{-1}$  konstruiert.

Ein Simplexschritt, der von einer nicht degenerierten Basislösung ausgeht, führt auf jeden Fall zu einer neuen Basislösung und damit auch zu einer neuen Ecke des Polyeders mit besserem Zielfunktionswert. Ist dagegen die Basislösung degeneriert, kann es passieren, dass die neue Basis nach dem Simplexschritt zur selben Basislösung und

damit auch zur selben Ecke gehört, so dass der Zielfunktionswert sich nicht ändert. Bei unvorsichtiger Wahl der Pivotelemente kann es zu sogenannten *Zyklen* kommen, bei der reihum immer wieder dieselben Basen besucht werden, so dass der Algorithmus in eine Endlosschleife läuft. Dies lässt sich aber beispielsweise durch eine geeignete Auswahl der Pivotzeile verhindern. In der Praxis ist eine weitere Methode, mit Zykeln umzugehen, eine absichtliche Perturbation (numerische Störung) der Daten, die nach einigen Iterationen wieder rausgerechnet wird, wenn der Algorithmus die betreffende Ecke verlassen hat.

Für die Wahl des Pivotelements gibt es meist mehrere Möglichkeiten. Die ursprünglich von Dantzig vorgeschlagene Methode wählt eine der Spalten mit dem größten reduzierten Kostenwert und eine beliebige Zeile, bei der nach dem Simplexschritt wieder eine zulässige (insbesondere nichtnegative) Basislösung entsteht. Dazu muss bei gegebener Pivotspalte  $s$  eine Pivotzeile gewählt werden, bei der das Minimum

$$\lambda := \min \left\{ \frac{\bar{b}_i}{\bar{a}_{is}} \mid i = 1, \dots, m, \bar{a}_{is} > 0 \right\}$$

angenommen wird. Sind alle Einträge in Spalte  $s$  negativ, ist das Optimierungsproblem unbeschränkt, da man Lösungen mit beliebig gutem Zielfunktionswert konstruieren könnte. In diesem Fall kann man aufhören.

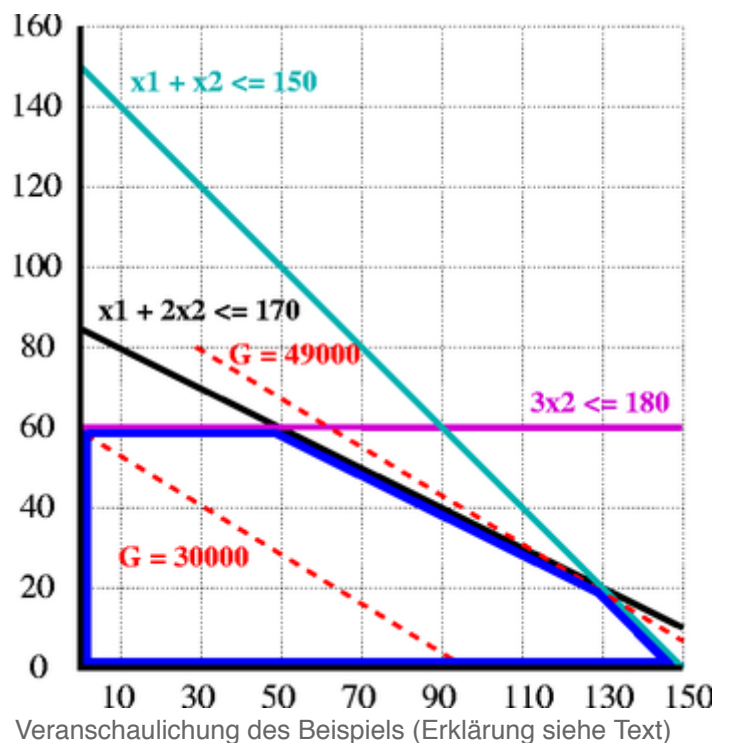
Im Normalfall gibt es sowohl mehrere Spalten mit positiven reduzierten Kosten als auch mehrere Zeilen, bei denen das Minimum  $\lambda$  angenommen wird, so dass eine Wahlmöglichkeit besteht. Da die Wahl des Pivotelements einen großen Einfluss auf die Rechenzeit haben kann, sind innerhalb der letzten 60 Jahre zahlreiche verbesserte Pivotstrategien gegenüber der Lehrbuchvariante entwickelt worden (siehe unten).

## Beispielrechnung

Anhand eines einfachen Beispiels zur Produktionsplanung mit zwei Variablen soll der Lösungsweg Schritt für Schritt gezeigt werden. In diesem einfachen Fall ist eine Optimallösung leicht zu finden. Reale Probleme können dagegen leicht aus mehreren Hunderttausend Variablen und Nebenbedingungen bestehen, so dass man ihnen meistens die Existenz einer Lösung oder den Wert einer Optimallösung nicht unmittelbar ansehen kann.

Eine Firma stellt zwei verschiedene Produkte her. Es stehen drei Maschinen A, B, C zur Verfügung. Maschine A hat eine maximale monatliche Laufzeit (Kapazität) von 170 Stunden, Maschine B von 150 Stunden und Maschine C von 180 Stunden. Eine Mengeneinheit (ME) von Produkt 1 liefert einen Deckungsbeitrag von 300 Euro, eine ME von Produkt 2 dagegen 500 Euro. Fertigt man 1 ME von Produkt 1, benötigt man dafür 1 Stunde die Maschine A und zusätzlich 1 Stunde die Maschine B. 1 ME von Produkt 2 belegt sowohl 2 Stunden Maschine A, als auch 1 Stunde Maschine B und 3 Stunden Maschine C.

Daraus erhält man folgende Bedingungen:





$$1 \cdot x_1 + 2 \cdot x_2 \leq 170 \quad (\text{Maschine A})$$

$$1 \cdot x_1 + 1 \cdot x_2 \leq 150 \quad (\text{Maschine B})$$

$$0 \cdot x_1 + 3 \cdot x_2 \leq 180 \quad (\text{Maschine C})$$

$$x_1 \geq 0, x_2 \geq 0$$

Die Firma möchte den Deckungsbeitrag  $z$  maximieren:

$$\max z = 300 \cdot x_1 + 500 \cdot x_2$$

$z$  ist daher der sogenannte Zielfunktionswert.

Eventuelle Fixkosten sind unabhängig von der Produktionsmenge und können daher einfach am Ende der Berechnung vom Gesamtdeckungsbeitrag abgezogen werden, um den Gewinn zu erhalten.

## Überführung in Gleichungen

Für das Simplex-Verfahren werden die Ungleichungen zunächst in Gleichungen überführt. Dazu führt man so genannte Schlupfvariablen  $y_A$ ,  $y_B$  und  $y_C$  ein, welche die nicht genutzten Zeiten der einzelnen Maschinen darstellen. Offensichtlich dürfen die Schlupfvariablen nicht negativ sein. Damit lässt sich das Problem so formulieren, dass man die Schlupfvariablen bezüglich der ursprünglichen Variablen freilegt. Ein derart genormtes Gleichungssystem heißt im Englischen *dictionary* (Benennung von V. Chvátal):

Maximiere die Zielfunktion  $z$  unter den Nebenbedingungen:

$$z = 0 + 300 x_1 + 500 x_2$$

$$y_A = 170 - 1 x_1 - 2 x_2$$

$$y_B = 150 - 1 x_1 - 1 x_2$$

$$y_C = 180 + 0 x_1 - 3 x_2$$

$$x_1 \geq 0, x_2 \geq 0$$

$$y_A \geq 0, y_B \geq 0, y_C \geq 0$$

Die obigen Gleichungen kann man in das vorher beschriebene *Simplex-Tableau* übertragen:

	$x_1$	$x_2$	rechte Seite
$-z$	300	500	0
$y_A$	1	2	170 = $b_1$
$y_B$	1	1	150 = $b_2$
$y_C$	0	3	180 = $b_3$

In der Kopfzeile stehen die *Nichtbasisvariablen*  $x_1$ ,  $x_2$  mit dem Wert 0, während die *Basisvariablen*  $y_i$  in der ersten Spalte stehen. In der obersten Zeile – der Gleichung für die Zielfunktion – stehen die *Zielfunktionskoeffizienten*  $c_i$ , also 300 und 500. Auf der rechten Seite steht die aktuelle Basislösung, was in diesem Fall genau  $b$  ist. In der obersten Zeile der rechten Seite steht das Negative des Zielfunktionswertes für die aktuelle Basislösung, im Beispiel also das Negative des Gesamtdeckungsbeitrags. Dieser ist momentan 0, da nichts produziert wird.

## Bestimmung einer zulässigen Ausgangslösung (Phase I)

Da die konstanten Größen des obigen Gleichungssystems nicht negativ sind, kann man die unabhängigen Variablen des Gleichungssystems (Nichtbasisvariablen) auf Null setzen und so eine triviale zulässige Lösung angeben, mit der direkt die Phase II gestartet werden kann:

$$\mathbf{x}_1 = 0, \mathbf{x}_2 = 0 \quad \mathbf{y}_A = 170, \mathbf{y}_B = 150, \mathbf{y}_C = 180$$

Die Variablen  $\mathbf{y}_i$  bilden eine zulässige Basis  $\mathbf{B}$ , deren Basismatrix  $\mathbf{A}_B$  die Einheitsmatrix ist. Die zugehörige Basislösung ist also  $\mathbf{A}_B^{-1}\mathbf{b} = \mathbf{b}$ . Diese Lösung entspricht dem Fall, dass nichts produziert wird ( $\mathbf{x}_i = 0$ ). Die Restkapazität der Maschinen, die durch die Werte der  $\mathbf{y}_i$  angegeben wird, ist hier deren Gesamtkapazität (170, 150 und 180), da die Maschinen nicht genutzt werden.

## Verbesserung der Ausgangslösung (Phase II)

Da die soeben berechnete Ausgangslösung unbefriedigend ist, wird nun in Phase II versucht, bessere zulässige Lösungen zu finden.

### Auswahl einer Pivotspalte und -zeile

In einer *Simplex-Iteration* wird eine *Basisvariable* gegen eine der unabhängigen Variablen ausgetauscht. In Frage kommen Nichtbasisvariablen mit positivem Zielfunktionskoeffizienten, da deren Aufnahme in die Basis den Zielfunktionswert verbessern kann. Diese Variable soll soweit erhöht werden, bis eine oder mehrere der Basisvariablen auf 0 stoßen. Eine beliebige dieser Basisvariablen verlässt dann die Basis und wird gegen die Nichtbasisvariable ausgetauscht.

Variable  $\mathbf{x}_1$  hat den positiven Zielfunktionskoeffizienten 300; indem sie erhöht wird, lässt sich die Zielfunktion  $\mathbf{z}$  vergrößern; sie kommt also als basis-eintretende Pivotvariable in Frage. Solange  $\mathbf{x}_1$  die *einzige* erhöhte Nichtbasisvariable ist, soll diese Erhöhung  $\bar{\mathbf{x}}_1$  durch folgende Bedingungen eingeschränkt werden:

$$0 \leq \bar{\mathbf{y}}_A = 170 - 1 \bar{\mathbf{x}}_1 \quad (\text{zu Basisvariable } \mathbf{y}_A)$$

$$0 \leq \bar{\mathbf{y}}_B = 150 - 1 \bar{\mathbf{x}}_1 \quad (\text{zu Basisvariable } \mathbf{y}_B)$$

In anderen Worten,

$$\bar{\mathbf{x}}_1 \leq \frac{170}{1}, \quad \bar{\mathbf{x}}_1 \leq \frac{150}{1} \quad \text{oder} \quad \bar{\mathbf{x}}_1 \leq \min \left\{ \frac{170}{1}, \frac{150}{1} \right\} = \min\{170, 150\} = 150$$

Die erste der Basisvariablen, die in diesem Fall auf Null stößt, erhält man über den Quotienten  $150/1$  und ist  $\mathbf{y}_B$ . Diese ist die Variable, die gegebenenfalls gegen  $\mathbf{x}_1$  ausgetauscht werden müsste. Der neue Wert der Zielfunktion wäre dann  $\bar{\mathbf{z}} = 300 \cdot (150/1) = 45000$ .

Auch Variable  $\mathbf{x}_2$  hat mit 500 einen positiven Zielfunktionskoeffizienten, kommt also ebenfalls als eintretende Nichtbasisvariable in Frage. Nach der obigen Vorgehensweise erhalten wir

$$0 \leq \bar{\mathbf{y}}_A = 170 - 2 \bar{\mathbf{x}}_2 \quad (\text{zu Basisvariable } \mathbf{y}_A)$$

$$0 \leq \bar{\mathbf{y}}_B = 150 - 1 \bar{\mathbf{x}}_2 \quad (\text{zu Basisvariable } \mathbf{y}_B)$$

$$0 \leq \bar{\mathbf{y}}_C = 180 - 3 \bar{\mathbf{x}}_2 \quad (\text{zu Basisvariable } \mathbf{y}_C)$$

und somit

$$\bar{\mathbf{x}}_2 \leq \min \left\{ \frac{170}{2}, \frac{150}{1}, \frac{180}{3} \right\} = \min\{85, 150, 60\} = 60$$

Der minimale Quotient  $(180/3)$  gehört zur Basisvariable  $y_C$ . Der neue Wert der Zielfunktion wäre  $\bar{z} = 500 \cdot (180/3) = 30000$ .

Für den Zuwachs der Zielfunktion in diesem Schritt ist es also am günstigsten, den ersten Fall zu wählen und die unabhängige Variable  $x_1$  anstelle der Basisvariablen  $y_B$  freizulegen.

### Durchführung eines Austauschschrittes

Mit dem Austauschschritt wird die Basisvariable  $y_B$  gegen die Nichtbasisvariable  $x_1$  ausgetauscht. Zuerst legen wir in der Gleichung für  $y_B$  die unabhängige Unbekannte  $x_1$  frei,

$$\begin{aligned} y_B &= 150 - 1x_1 - 1x_2 \\ \implies x_1 &= 150 - 1y_B - 1x_2 \end{aligned}$$

und anschließend ersetzen wir  $x_1$  in den restlichen Gleichungen für den so erhaltenen Ausdruck:

$$\begin{aligned} z &= 0 + 300(150 - 1y_B - 1x_2) + 500x_2 \\ y_A &= 170 - 1(150 - 1y_B - 1x_2) - 2x_2 \\ y_C &= 180 + 0(150 - 1y_B - 1x_2) - 3x_2 \end{aligned}$$

Das führt zu den oben beschriebenen Verwandlungsregeln für das Simplex-Tableau nach dem Pivotelement  $a_{21} = 1$ . Wenn  $x_1$  die Zeile von  $y_B$  besetzt und  $y_B$  die Spalte von  $x_1$ , dann ist das neue Gleichungssystem

$$\begin{aligned} z &= 45000 - 300y_B + 200x_2 \\ y_A &= 20 + 1y_B - 1x_2 \\ x_1 &= 150 - 1y_B - 1x_2 \\ y_C &= 180 + 0y_B - 3x_2 \end{aligned}$$

Die Unbekannte  $x_1$  ist in die Basis gerückt, die jetzt unabhängige Unbekannte  $y_B$  ist eine Nichtbasisvariable und erscheint in der Kopfzeile.

Diese Lösung bedeutet: Es werden **150** ME von Produkt 1 (Gleichung mit freigelegtem  $x_1$ ) produziert. Von Produkt 2 wird nichts gefertigt ( $x_2 = 0$  ist Nichtbasisvariable). Damit erzielt die Firma einen Gesamtdeckungsbeitrag von 45.000 Euro. Maschine A steht **20** Stunden pro Monat still (sie läuft nur 150 der 170 möglichen Stunden). Maschine B hat keine Leerlaufzeit. Maschine C steht **180** Stunden still, das heißt, sie wird überhaupt nicht benötigt. Dies ergibt sich auch direkt aus der Aufgabenstellung: Maschine C wird nur bei der Herstellung von Produkt 2 benötigt. Da dieses nicht gefertigt wird, braucht man Maschine C noch nicht.

Das zum obigen Gleichungssystem gehörende neue Simplex-Tableau ist

	$y_B$	$x_2$	rechte Seite
$-z$	-300	200	-45000
$y_A$	-1	1	20 = $b_1$
$x_1$	1	1	150 = $b_2$
$y_C$	0	3	180 = $b_3$

Die Einträge des neuen Simplex-Tableaus können anhand der oben angeführten Formeln errechnet werden.

### Wiederholung der Simplexschritte

Da die Zielfunktion im entstandenen Simplex-Tableau noch einen positiven Koeffizienten enthält, kann man die Lösung noch verbessern. Dies geschieht mittels einer weiteren Simplex-Iteration. Bei der Auswahl des Pivotelementes kommt nur die Unbekannte  $x_2$  in Frage, da nur hier der Zielfunktionskoeffizient positiv ist. Die Basisvariable des Pivots ergibt sich aus

$$0 \leq \bar{y}_A = 20 - 1 \bar{x}_2 \quad (\text{zu Basisvariable } y_A)$$

$$0 \leq \bar{x}_1 = 150 - 1 \bar{x}_2 \quad (\text{zu Basisvariable } x_1)$$

$$0 \leq \bar{y}_C = 180 - 3 \bar{x}_2 \quad (\text{zu Basisvariable } y_C)$$

und somit

$$\bar{x}_2 \leq \min \left\{ \frac{20}{1}, \frac{150}{1}, \frac{180}{3} \right\} = \min\{20, 150, 60\} = 20$$

Damit ist Zeile  $y_A$  die einzige Basisunbekannte, die für einen Pivot in Frage kommt. Die Basisvariable  $y_A$  wird gegen die Nichtbasisvariable  $x_2$  ausgetauscht; das Pivotelement ist **1**. Mit den gleichen Umrechnungen wie oben erhält man als neues Gleichungssystem

$$z = 49000 - 100 y_B - 200 y_A$$

$$x_2 = 20 + 1 y_B - 1 y_A$$

$$x_1 = 130 - 2 y_B + 1 y_A$$

$$y_C = 120 - 3 y_B + 3 y_A$$

beziehungsweise ein neues Simplex-Tableau

	$y_B$	$y_A$	rechte Seite
$-z$	-100	-200	-49000
$x_2$	-1	1	20
$x_1$	2	-1	130
$y_C$	3	-3	120

Da die Zielfunktion nun keine positiven Koeffizienten mehr enthält, ist eine optimale Lösung erreicht. Es werden **130** ME von Produkt 1 und **20** ME von Produkt 2 hergestellt. Damit erzielt die Firma einen Gesamtdeckungsbeitrag von 49.000 Euro. Maschine A und Maschine B sind voll ausgelastet. Maschine C läuft 60 Stunden und hat daher noch eine ungenutzte Kapazität von  $y_C = 120$  Stunden.

## Einträge in Bruchzahlform

Das obige Beispiel wurde in algebraischer Notation gelöst, indem man im Gleichungssystem die Basisvariablen bezüglich der restlichen, unabhängigen Variablen freilegt. Um Rundungsfehler zu vermeiden, kann man mit Bruchzahlen arbeiten und *einen* gemeinsamen Nenner für das gesamte Gleichungssystem wählen (dass dieser Nenner oben immer **1** war, ist Zufall). Um in jedem Schritt den gemeinsamen Nenner für das Gesamtsystem zu finden, müssen wir die Einträge *nicht* zusätzlich analysieren. Falls das Startsystem ganzzahlig ist (was sich normalerweise durch Erweiterung erreichen lässt), gilt die Regel:

- Der Zähler des gewählten Pivotelements ist ein gemeinsamer Nenner für das darauffolgende System

Wenn wir die neuen Tableau-Einträge mit diesem gemeinsamen Nenner multiplizieren, erhalten wir stets ganzzahlige Zähler. Bei der Berechnung dieser Zähler für die neuen Tableau-Einträge wird dann ungeprüft durch den *alten* Nenner *geteilt*, wobei das Ergebnis ganzzahlig sein muss.

Als Beispiel für diese Vorgehensweise lösen wir dieselbe Aufgabe wie oben noch einmal mit *Dantzig's Pivotauswahl*; hierbei wird als eingehende Pivotvariable diejenige mit größtem Zielfunktionskoeffizienten gewählt:

$$\begin{aligned} z &= ( 0 + 300 x_1 + 500 x_2 ) / 1 \\ y_A &= ( 170 - x_1 - 2 x_2 ) / 1 \\ y_B &= ( 150 - x_1 - x_2 ) / 1 \\ y_C &= ( 180 + 0 x_1 - 3 x_2 ) / 1 \end{aligned}$$

Durch Erhöhung der unabhängigen Variable  $x_2$  lässt sich die Zielfunktion  $z$  vergrößern; die erste der Basisvariablen, die in diesem Fall auf Null stößt, ist  $y_C$ . Folglich kann man  $x_2$  anstelle von  $y_C$  freilegen und erhält folgendes System mit  $\bar{z} = 30000$ :

$$\begin{aligned} z &= ( 90000 + 900 x_1 - 500 y_C ) / 3 \\ y_A &= ( 150 - 3 x_1 + 2 y_C ) / 3 \\ y_B &= ( 270 - 3 x_1 + y_C ) / 3 \\ x_2 &= ( 180 + 0 x_1 - y_C ) / 3 \end{aligned}$$

Wenn man die unabhängige Variable  $x_1$  erhöht, vergrößert man die Zielfunktion; die erste der Basisvariablen, die dann auf Null stößt, ist  $y_A$ . Folglich legt man  $x_1$  anstelle von  $y_A$  frei und erhält folgendes System mit  $\bar{z} = 45000$ :

$$\begin{aligned} z &= ( 135000 - 900 y_A + 100 y_C ) / 3 \\ x_1 &= ( 150 - 3 y_A + 2 y_C ) / 3 \\ y_B &= ( 120 + 3 y_A - y_C ) / 3 \\ x_2 &= ( 180 + 0 y_A - y_C ) / 3 \end{aligned}$$

Wenn man die unabhängige Variable  $y_C$  erhöht, vergrößert man die Zielfunktion; die erste der Basisvariablen, die dann auf Null stößt, ist  $y_B$ . Folglich legt man  $y_C$  anstelle von  $y_B$  frei und erhält folgendes System mit  $\bar{z} = 49000$ :

$$\begin{aligned} z &= ( 49000 - 200 y_A - 100 y_B ) / 1 \\ x_1 &= ( 130 + y_A - 2 y_B ) / 1 \\ y_C &= ( 120 + 3 y_A - 3 y_B ) / 1 \\ x_2 &= ( 20 - y_A + y_B ) / 1 \end{aligned}$$

Die Zielfunktion hat Wert  $\bar{z} = 49000$  und lässt sich nicht mehr erhöhen; die Lösung ist wie oben  $\bar{x}_1 = 130$ ,  $\bar{x}_2 = 20$ . Wir beobachten nebenher, dass Dantzig's Pivotauswahl im Vergleich zur anfangs gewählten Alternative einen Schritt mehr gebraucht hat, um zur Lösung zu gelangen.

## Beispielhafte Lösung in Java

Das Simplex-Verfahren steht bereits als Software-Implementierung in zahlreichen Software-Bibliotheken zur Verfügung. Ein Beispiel ist z. B. das Java-Package *org.apache.commons.math3.optim* von Apache Commons™. Die zu maximierende Funktion wird als *LinearObjectiveFunction* (Zeile: 22) definiert und entspricht dem Deckungsbeitrag

aus dem obigen Beispiel. Die Bedingungen werden als *LinearConstraint* (Zeile: 24–26) angegeben. Der *SimplexSolver* (Zeile: 28) benötigt zur Lösung die Anzahl an maximalen Iterationen, die zu maximierende Funktion, die Bedingungen und die Information ob ein Maximum bzw. Minimum gesucht wird (Zeile: 29). Zusätzlich erhält der *SimplexSolver* mit dem *NonNegativeConstraint* die Information, dass für die Koordinaten  $x_1, x_2$  Werte größer 0 gesucht werden.

```

1 package SimplexAlgorithmExample;
2
3 import java.util.ArrayList;
4 import java.util.Arrays;
5 import java.util.Collection;
6
7 import org.apache.commons.math3.optim.MaxIter;
8 import org.apache.commons.math3.optim.PointValuePair;
9 import org.apache.commons.math3.optim.linear.LinearConstraint;
10 import org.apache.commons.math3.optim.linear.LinearConstraintSet;
11 import org.apache.commons.math3.optim.linear.LinearObjectiveFunction;
12 import org.apache.commons.math3.optim.linear.NonNegativeConstraint;
13 import org.apache.commons.math3.optim.linear.Relationship;
14 import org.apache.commons.math3.optim.linear.SimplexSolver;
15 import org.apache.commons.math3.optim.nonlinear.scalar.GoalType;
16
17
18 public class SimplexAlgorithmExample
19 {
20     public static void main(String[] args)
21     {
22         LinearObjectiveFunction oFunc = new LinearObjectiveFunction(new double[] {300, 500}, 0);
23         Collection<LinearConstraint> constraints = new ArrayList<LinearConstraint>();
24         constraints.add(new LinearConstraint(new double[] {1, 2}, Relationship.LEQ, 170));
25         constraints.add(new LinearConstraint(new double[] {1, 1}, Relationship.LEQ, 150));
26         constraints.add(new LinearConstraint(new double[] {0, 3}, Relationship.LEQ, 180));
27
28         SimplexSolver solver = new SimplexSolver();
29         PointValuePair solution = solver.optimize(new MaxIter(100), oFunc, new
LinearConstraintSet(constraints), GoalType.MAXIMIZE, new NonNegativeConstraint(true));
30         System.out.println(Arrays.toString(solution.getPoint()) + " : " + solution.getSecond());
31     }
32 }

```

## Duale Information im Tableau

Aus dem Simplextableau lässt sich auch die Information zur Lösung des zu dem linearen Programm (LP) gehörigen dualen linearen Programms entnehmen. Zu einer gegebenen Basis  $B$  kann man neben der zugehörigen *Primallösung*  $x = \bar{b} = A_B^{-1}b$ , die in der rechten Spalte des Tableaus steht, auch eine *Duallösung*  $\pi^T = c_B^T A_B^{-1}$  berechnen. Diese beiden Vektoren  $x$  und  $\pi$  erfüllen immer die Bedingung

$$\pi^T b = c^T x,$$

die für Optimallösungen notwendig ist. Der Vektor  $x$  bleibt nach Konstruktion der einzelnen Simplexiterationen immer zulässig für das primale LP, während der Vektor  $\pi$  Bedingungen des dualen LPs verletzen kann. Sind zu einer Basis sowohl die zugehörige Primallösung  $x$  als auch die entsprechende Duallösung  $\pi$  zulässig (man spricht dann von einer *primal und dual zulässigen Basis*), dann sind beide optimal für das primale bzw. duale lineare Programm. Man kann zeigen, dass dies genau dann der Fall ist, wenn der reduzierte Kostenvektor  $\bar{c}$  nichtpositiv ist. Obwohl das Ziel des Simplex-Verfahrens eigentlich nur die Berechnung einer optimalen Primallösung ist, fällt also am Ende auch eine optimale Duallösung nebenbei mit ab.

Eine Beispielrechnung zur Dualität befindet sich im Artikel [Pivotverfahren](#).

## Varianten und Verbesserungen

In der hier vorgestellten Form, die im Wesentlichen der ursprünglichen Version von Dantzig entspricht, wird der Simplex-Algorithmus in praktischen Implementierungen heute nicht mehr verwendet. Im Laufe der Zeit sind einige Varianten des Simplex-Verfahrens entwickelt worden, die die Rechenzeit und den Speicherbedarf beim Lösen linearer Programme gegenüber dem Standardverfahren deutlich verkürzen und numerisch deutlich stabiler sind. Die wichtigsten Verbesserungen, die heute zum Standard in guten LP-Lösern gehören, sollen hier kurz vorgestellt werden.

## Auswahl des Pivotelementes

Bei der Auswahl des Pivotelements hat man in der Regel einige Freiheiten. Die Pivotspalte  $s$  und die Pivotzeile  $r$  können beliebig gewählt werden, unter den Bedingungen, dass

- die Pivotspalte positive reduzierte Kosten hat und
- die Pivotzeile wieder zu einer zulässigen Basislösung führt.

Die Wahl des Pivotelementes hat oft großen Einfluss auf die Anzahl der Iterationen und auf die numerische Stabilität des Verfahrens, insbesondere bei degenerierten Problemen. Die Entwicklung besserer Pivotstrategien, insbesondere zur Spaltenauswahl, haben im Laufe der Zeit große Fortschritte bei der Beschleunigung des Lösungsprozesses bewirkt.

### Spaltenauswahl

Für die Auswahl der Pivotspalte (engl. *pricing*) gibt es verschiedene Strategien, die unterschiedlichen Rechenaufwand erfordern und je nach Eingabedaten unterschiedlich gut funktionieren:<sup>[5][6]</sup>

- Wähle die erste geeignete Spalte. Dies ist die einfachste Variante, die aber oft zu sehr vielen Iterationen führt und daher in der Praxis nicht verwendet wird.
- Die ursprünglich von Dantzig vorgeschlagene Methode wählt eine der Spalten mit dem größten reduzierten Kostenwert. Diese Variante kann bei vielen Variablen viel Rechenzeit beanspruchen.
- Das *steepest-edge pricing* ist eine Kombination aus Spalten- und Zeilenwahl, die zusammen den größten Fortschritt für die Zielfunktion bringen. Diese Variante ist in jeder Iteration sehr aufwändig, führt aber oft zu wenigen Iterationen.
- Das *devex pricing* ist eine 1974 von Paula Harris vorgeschlagene Approximation von *steepest-edge pricing* und eines der Standardverfahren in heutigen LP-Lösern. Hierbei werden die Spalten der Matrix und die reduzierten Kosten vor der Auswahl auf eine einheitliche Norm skaliert, um die Aussagekraft der reduzierten Kosten zu erhöhen.
- Beim *partial pricing* wird die Variablenmenge in Blöcke unterteilt und eines der obigen vorherigen Verfahren auf einen Block angewendet. Erst wenn dort keine geeignete Variable gefunden wird, wird überhaupt der nächste Block betrachtet.
- Das *multiple pricing* sucht einmal eine Menge von geeigneten Variablen heraus, die dann in den nächsten Iterationen bevorzugt als Kandidaten betrachtet werden. Erst wenn keiner dieser Kandidaten mehr positive reduzierte Kosten besitzt, werden die anderen Variablen betrachtet.
- Das *partial multiple pricing* ist eine Kombination der letzten beiden Varianten, die neue Kandidaten immer nur aus einem Teil aller zur Verfügung stehenden Variablen bestimmt. Diese Strategie gehört neben *devex pricing* heute zu den Standardstrategien.
- Beim *hybrid pricing* werden mehrere Strategien je nach Situation abwechselnd verwendet. Einige LP-Löser wenden zusätzlich noch numerische Kriterien bei der Spaltenauswahl an, um die Probleme durch Rundungsfehler in Grenzen zu halten.
- Die Regel von R. G. Bland<sup>[7]</sup> wählt zunächst die Spalte mit kleinstem Index unter allen in Frage kommenden Spalten. Das ist sinnvoll, wenn danach bei mehreren zur Auswahl stehenden Zeilen ebenfalls diejenige mit kleinstem Index gewählt wird. Ein Beweis, dass diese Regel Zyklen verhindert, ist zum Beispiel in<sup>[8]</sup> enthalten.

### Zeilenauswahl

Gibt es mehrere geeignete Pivotzeilen, hat man die Wahl zwischen mehreren Varianten:

- Wähle die erste geeignete Zeile. Diese Variante ist zwar pro Iteration sehr schnell, führt aber insgesamt oft zu vielen Iterationen und ist numerisch instabil.
- Die *lexikographische Auswahlregel* wählt unter allen in Frage kommenden Zeilen die (eindeutige) lexikographisch kleinste Zeile aus. Diese Regel ist unter dem Gesichtspunkt der Geschwindigkeit nicht besonders gut, verhindert aber, dass Tableaus mehrfach besucht werden und der Algorithmus ins Zykeln gerät. Aus diesem Grund kann sie

beispielsweise für einige Iterationen verwendet werden, um von einer Basislösung wegzukommen, bevor wieder auf andere Auswahlverfahren umgestellt wird.

- Der 1973 von Paula Harris vorgeschlagene *Harris-Quotiententest*, der heute zu den Standardverfahren zählt, erlaubt aus Gründen der numerischen Stabilität eine leichte Unzulässigkeit der neuen Lösung.
- Wähle unter den geeigneten Zeilen zufällig. Zyklen werden so sehr unwahrscheinlich, aber nicht unmöglich.

## Variablenschranken

In der Praxis müssen häufig obere und untere Schranken für die Variablen berücksichtigt werden. Dies gilt insbesondere dann, wenn lineare Programme beispielsweise im Rahmen eines Branch-and-Cut-Prozesses als Unterproblem gelöst werden. Für solche einfachen Arten von Ungleichungen wie Variablenschranken gibt es das sogenannte *Bounded-Simplex-Verfahren*, bei dem die Schranken direkt in den einzelnen Simplex-Schritten berücksichtigt werden. Im Gegensatz zur Standardversion, bei dem eine Nichtbasisvariable immer den Wert 0 hat, kann sie jetzt auch den Wert einer ihrer Schranken annehmen. Diese Mitführung der Schranken in den einzelnen Schritten bewirkt eine kleinere Anzahl der Zeilen und damit eine kleinere Basis gegenüber der offensichtlichen Variante, Variablenschranken als Ungleichungen in das LP zu schreiben.

## Duales Simplex-Verfahren

Neben einer optimalen *Primallösung*, also einer Lösung für das gegebene lineare Programm, berechnet das oben beschriebene *primale Simplex-Verfahren* auch eine optimale *Duallösung*, also eine Lösung des zugehörigen dualen linearen Programms. Da das duale LP aus dem primalen im Wesentlichen durch Vertauschung von Zeilen und Spalten entsteht, lässt sich mit dem Simplex-Verfahren auch das duale Problem lösen, indem man das gegenüber der obigen Variante leicht modifizierte *Tucker-Tableau* verwendet und im beschriebenen Algorithmus Zeilen und Spalten vertauscht. Diese Variante heißt dann *duales Simplex-Verfahren*. Es wurde erstmals 1954 von Carlton Lemke und E. M. L. Beale beschrieben, ist aber erst seit Anfang der 1990er Jahre fester Bestandteil kommerzieller LP-Löser, als erfolgreiche Pivotstrategien dafür entwickelt wurden, wie das *dual steepest-edge pricing* in der Version von Forrest und Goldfarb aus dem Jahre 1992.

In der Praxis hängt die Laufzeit des Simplex-Verfahrens oft im Wesentlichen von der Anzahl der Zeilen im LP ab. Dies gilt insbesondere für dünnbesetzte Matrizen, wie sie in der Praxis normalerweise auftreten. Wenn ein lineares Programm sehr viele Bedingungen, aber nur wenige Variablen hat, kann es sich daher lohnen, stattdessen das duale LP zu lösen, bei dem Zeilen- und Spaltenzahl vertauscht sind, und sich dabei eine optimale Primallösung mitliefern zu lassen, die nebenbei mitberechnet wird.

Ein weiterer großer Vorteil der primal-dualen Betrachtungsweise ist es, dass primale und duale Simplexschritte abwechselnd im selben Tableau durchgeführt werden können. Anstatt das Tableau explizit zu transponieren, werden einfach im Algorithmus Zeilen- und Spaltenoperationen vertauscht, je nachdem, ob gerade der primale oder der duale Algorithmus benutzt wird. Im Gegensatz zum primalen Simplex-Verfahren, das immer eine zulässige Primallösung behält und erst am Ende eine zulässige Duallösung erreicht, ist es beim dualen Simplex-Verfahren umgekehrt. Wenn also die Primallösung zu einer Basis unzulässig, die zugehörige Duallösung aber zulässig ist, kann man durch duale Simplexschritte versuchen, wieder zu einer primal zulässigen Lösung zu kommen. Dies kann man in mehreren Zusammenhängen ausnutzen, die hier kurz beschrieben werden sollen.

- Im Verlauf von Schnittebenenverfahren oder Branch-and-Cut-Verfahren wird sehr oft in einem gerade gelösten LP eine Variablenschranke verändert oder eine Ungleichung hinzugefügt, die von der alten Lösung nicht erfüllt wird, und anschließend das LP neu gelöst. Da die alte Basislösung jetzt nicht mehr zulässig ist, ist eine der Grundbedingungen des primalen Simplextableaus verletzt, so dass das primale Simplexverfahren von vorne starten muss, um das neue LP zu lösen. Wenn an der Zielfunktion nichts verändert wurde, ist aber die alte Duallösung weiter zulässig, so dass mit einigen dualen Simplexschritten von der alten Startbasis aus meist nach wenigen Iterationen eine Optimallösung für das modifizierte LP gefunden wird. Dieser Unterschied schlägt sich bei großen LPs oft sehr deutlich in der Gesamtlaufzeit nieder.
- Wenn im Verlauf des Algorithmus numerische Schwierigkeiten auftreten oder es sehr lange keinen Fortschritt in der Zielfunktion gibt, kann es sinnvoll sein, vorübergehend eine leichte Verletzung von Variablenschranken zu



erlauben, um sich aus einer kritischen Ecke des Polytops hinauszubewegen. Dies kann anschließend mit einigen dualen Simplexschritten wieder behoben werden.

- Wenn das lineare Programm bestimmte Strukturen aufweist, kann man direkt eine primal unzulässige, aber dual zulässige Startbasis angeben, ohne dafür rechnen zu müssen. In solch einem Fall kann man von dieser Basis aus direkt in Phase II mit dualen Simplexschritten starten und kann sich die Phase I sparen.

## Revidiertes Simplex-Verfahren

Obwohl praktisch auftretende lineare Programme mehrere hunderttausend Variablen haben können, arbeitet das Simplex-Verfahren immer nur mit einem kleinen Teil davon, nämlich den Basisvariablen. Lediglich bei der Spaltenauswahl müssen die Nichtbasisspalten betrachtet werden, wobei es – je nach Pivotstrategie – oft ausreicht, nur einen Teil davon zu berücksichtigen. Diese Tatsache macht sich das *revidierte Simplex-Verfahren* zunutze, das immer nur die aktuelle Basismatrix  $A_B$  oder deren Inverse speichert, zusammen mit etwas Zusatzinformationen, aus der die aktuelle Basismatrix bzw. deren Inverse berechnet werden kann. Dadurch kommt es mit wesentlich weniger Speicherplatz aus als das ursprüngliche Tableauverfahren. Dieses Verfahren bildet heute die Grundlage mehrerer guter LP-Löser.

In der ersten kommerziellen Implementierung dieses Verfahrens von William Orchard Hays im Jahre 1954 wurde die Basisinverse noch in jedem Schritt komplett neu berechnet, was mit den damaligen Lochkartenrechnern eine Herausforderung darstellte. Wenig später implementierte er die sogenannte *Produktform der Inversen* nach einer Idee von A. Orden. Dabei wurde nur die erste Basisinverse gespeichert, zusammen mit Update-Vektoren, aus denen die aktuelle Inverse in jedem Schritt berechnet werden konnte. Der damalige Rekord lag bei einem linearen Programm mit 71 Variablen und 26 Ungleichungen, das innerhalb von acht Stunden optimal gelöst wurde. In heute verwendeten Implementierungen wird eher die Basismatrix selbst mit Hilfe einer speziellen Form der LR-Zerlegung gespeichert, da die Inverse einer dünnbesetzten Matrix in der Regel nicht dünnbesetzt ist.

## LR-Zerlegungen

Im Verlauf des Simplex-Verfahrens werden sehr viele ähnliche lineare Gleichungssysteme gelöst. Da große lineare Gleichungssysteme auch in anderen Zusammenhängen (beispielsweise bei der Lösung von Differentialgleichungen) häufig auftreten, wurden in den 1970er Jahren in der numerischen linearen Algebra Verfahren zur LR-Zerlegung einer Matrix entwickelt. Dabei wird die Matrix in eine untere und eine obere Dreiecksmatrix zerlegt, was es anschließend erlaubt, viele Gleichungssysteme mit derselben Matrix, aber unterschiedlichen rechten Seiten effizient zu lösen.

Im Falle des Simplex-Verfahrens wird diese Methode auf die Basismatrix  $A_B$  angewandt. Da diese sich im Laufe des Simplex-Algorithmus ständig ändert, muss ihre LR-Zerlegung bei jedem Simplexschritt angepasst werden, beispielsweise mit Hilfe des nach seinen Erfindern benannten *Forest-Tomlin-Updates*. Diese Anpassung erfordert nur sehr geringen Aufwand im Vergleich zur Berechnung einer komplett neuen LR-Zerlegung, weil sich die Basismatrix in jeder Iteration nur in einer Spalte ändert. In praktischen Implementierungen wird meist aus numerischen Gründen trotzdem alle paar hundert Iterationen eine komplett neue LR-Zerlegung der aktuellen Matrix berechnet. Oft kann die Matrix schon durch geschickte Umsortierung der Zeilen und Spalten größtenteils in Dreiecksform gebracht werden, so dass nur noch ein kleiner Teil der Basismatrix, der sogenannte *Nucleus*, tatsächlich faktorisiert werden muss. Für die Berechnung der LR-Zerlegung selbst gibt es wieder verschiedene Varianten, von denen sich in der Praxis vor allem die LR-Zerlegung mit *dynamischer Markowitz-Pivotisierung* durchgesetzt hat, da diese die Dünnbesetztheit von Matrizen bei der Zerlegung weitgehend erhält. Dieses Verfahren hat vor allem für große lineare Programme, die fast immer dünnbesetzt sind, zu starken Reduktionen der Rechenzeit geführt.

## Preprocessing

In den letzten zehn Jahren sind durch verbessertes Preprocessing sehr große Fortschritte in den Lösungszeiten erzielt worden. Beispielsweise gibt es oft numerische Probleme, wenn in einem linearen Gleichungssystem sowohl sehr große als auch sehr kleine Zahlen auftreten. In einigen Fällen lässt sich dies durch Vorkonditionierung, also z. B.

Äquilibrierung des Gleichungssystems, vor dem Start des eigentlichen Algorithmus vermeiden.

Eine andere Klasse von Methoden des Preprocessing versucht, Redundanzen im linearen Gleichungssystem zu erkennen oder Variablenschranken zu verschärfen, um die Anzahl der Zeilen und Spalten zu reduzieren:

- Wenn eine Zeile linear abhängig von anderen Zeilen ist, ist sie überflüssig und kann entfernt werden. Dies ist allerdings – bis auf den Spezialfall, dass eine Zeile ein skalares Vielfaches einer anderen Zeile ist – im Allgemeinen schwierig mit vertretbarem Aufwand zu erkennen.
- Sehr oft sind Variablen aufgrund von Bedingungen auf einen bestimmten Wertebereich beschränkt oder durch andere Variablen festgelegt. Beispielsweise sind durch die Gleichung  $x_1 + x_2 = 1$  und die Nichtnegativitätsbedingungen beide Variablen auf den Bereich  $[0, 1]$  beschränkt. Die Kenntnis dieser Schranke kann den Lösungsprozess beschleunigen. Darüber hinaus ist der Wert von  $x_2$  durch den Wert von  $x_1$  festgelegt. Dadurch kann man in allen Bedingungen, in denen  $x_2$  vorkommt, diese Variable durch  $1 - x_1$  ersetzen und  $x_2$  aus dem linearen Programm entfernen.
- Wenn mehrere Variablen auf einen bestimmten Wertebereich fixiert wurden, kann es sein, dass einige Ungleichungen immer erfüllt sind oder nicht mehr erfüllt werden können. Im ersten Fall kann die Ungleichung entfernt werden, im zweiten Fall ist sofort die Unzulässigkeit des linearen Programms gezeigt, und man kann aufhören.

Mit Hilfe solcher Methoden kann gerade bei sehr großen linearen Programmen die Anzahl der Zeilen und Spalten manchmal deutlich reduziert werden, was sich in sehr viel kürzeren Lösungszeiten widerspiegelt.

## Literatur


- George B. Dantzig: *Lineare Programmierung und Erweiterungen*. Springer-Verlag, 1966 (Originalausgabe: *Linear Programming and Extensions*, Princeton University Press, ISBN 0-691-05913-6).
- V. Klee, G.J. Minty: *How Good is the Simplex Algorithm?* In: O. Shisha (editor): *Inequalities III*. Academic Press, New York 1972, S. 159–175.
- Vašek Chvátal: *Linear Programming*. W. H. Freeman and Company, New York 1983, ISBN 0-7167-1587-2.
- Alexander Schrijver: *Theory of Linear and Integer Programming*. John Wiley and Sons, 1998, ISBN 0-471-98232-6.
- Wolfgang Domschke, Andreas Drexl: *Einführung in Operations Research*. 7. Auflage. Springer, Berlin 2007, ISBN 978-3-540-70948-0.
- Michael Sauer: *Operations Research kompakt* 1. Auflage. Oldenbourg, München 2009, ISBN 978-3-486-59082-1.
- Winfried Hochstättler: *Algorithmische Mathematik*. Springer, Berlin / Heidelberg 2010, ISBN 978-3-642-05421-1.

## Weblinks

- Hilfreiche OR-Zusammenfassung der TU Kaiserslautern ([https://www.wiwi.uni-kl.de/bisor-orwiki/Lineare\\_Planungsrechnung\\_und\\_Optimierung](https://www.wiwi.uni-kl.de/bisor-orwiki/Lineare_Planungsrechnung_und_Optimierung))
- simplex me – the simple simplex solver. (<http://www.simplexme.com/>) (Lineare Optimierungsprobleme mit Simplex-Algorithmus lösen – mehrsprachig, PDF Export)
- Simplex Applet (<http://algorithms.wtf/complexity.html>) mit textueller Eingabe des Programms
- Visualisierung der Lösungsschritte des Simplex-Algorithmus (<http://simplexsolver.jumland.de/>) (Java-Programm)
- Rechner Simplexalgorithmus (<https://www.matopt.de/werkzeuge/lineare-optimierung/simplexalgorithmus.html>) zeigt alle Zwischenschritte mit Erläuterung
- GNU scientific library (<http://www.gnu.org/software/gsl/>) enthält simplex code
- nonlinear optimization (<https://web.archive.org/web/20081229150811/http://ab-initio.mit.edu/wiki/index.php/NLOpt>) (Memento vom 29. Dezember 2008 im *Internet Archive*)

## Einzelnachweise

1. John A. Nelder, R. Mead: *A simplex method for function minimization*. In: *Computer Journal*. 7, 1965, S. 308–313. doi:10.1093/comjnl/7.4.308 (<https://doi.org/10.1093/comjnl%2F7.4.308>).
2. Robert Bixby: *Solving real-world linear programs: A decade and more of progress*. In: *Operations Research*, Band 50, Nr. 1, 2002, S. 3–15.
3. Harvey J. Greenberg: *Klee-Minty Polytope Shows Exponential Time Complexity of Simplex Method*. (<http://glossary.computing.society.informs.org/notes/Klee-Minty.pdf>) (PDF) University of Colorado at Denver, 1997.

4. Martin Grötschel: *Algorithmische Diskrete Mathematik II: Lineare Optimierung*, Vorlesungsskript (<http://www.zib.de/groetschel/teaching/skriptADMII.pdf>) (PDF)
5. István Maros: *A General Pricing Scheme for the Simplex Method*. (<http://citeseer.ist.psu.edu/rd/0%2C475411%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/23185/http:zSzzSzwww.doc.ic.ac.ukzSzdeptechrepzSzDTR01-3.pdf/maros01general.pdf>) (PDF) Technical Report 2001/3, Department of Computing, Imperial College, London 2001, ISSN 1469-4174.
6. Roland Wunderling: *Paralleler und Objektorientierter Simplex-Algorithmus*. (<https://web.archive.org/web/20060925105244/http://www.zib.de/Publications/abstracts/TR-96-09/>) (Memento des Originals (<https://tools.wmflabs.org/giftbot/deref.fcgi?url=http%3A%2F%2Fwww.zib.de%2FPublications%2Fabstracts%2FTR-96-09%2F>) vom 25. September 2006 im *Internet Archive*)  Info: Der Archivlink wurde automatisch eingesetzt und noch nicht geprüft. Bitte prüfe Original- und Archivlink gemäß Anleitung und entferne dann diesen Hinweis. Dissertation, TU Berlin, 1996.
7. R. Sedgewick: *Algorithmen*. Addison-Wesley Publishing Company, 1992, ISBN 3-89319-301-4, S. 697.
8. M. Padberg: *Linear Optimization and Extensions*. Springer, Berlin/Heidelberg 1995.

---

Abgerufen von „<https://de.wikipedia.org/w/index.php?title=Simplex-Verfahren&oldid=189257044>“

---

**Diese Seite wurde zuletzt am 4. Juni 2019 um 21:25 Uhr bearbeitet.**

Der Text ist unter der Lizenz „Creative Commons Attribution/Share Alike“ verfügbar; Informationen zu den Urhebern und zum Lizenzstatus eingebundener Mediendateien (etwa Bilder oder Videos) können im Regelfall durch Anklicken dieser abgerufen werden. Möglicherweise unterliegen die Inhalte jeweils zusätzlichen Bedingungen. Durch die Nutzung dieser Website erklären Sie sich mit den Nutzungsbedingungen und der Datenschutzrichtlinie einverstanden. Wikipedia® ist eine eingetragene Marke der Wikimedia Foundation Inc.