Lineare Optimierung anhand von Beispielen

Lineare Ungleichungssysteme zur Lösung von Optimierungsprozessen im Bereich Operations Research

Weiterführende Fassung der

VORWISSENSCHAFTLICHEN ARBEIT

im Bereich „Mathematik“

Bundesrealgymnasium
9800 Spittal/Drau, Zernattostraße 10

BETREUERIN
Prof. Mag. Thorbauer-Noisternig

VERFASSER
Markus Tripp, 8B

Spittal/Drau, am 14. Februar 2019

Abstract

Die vorliegende Vorwissenschaftliche Arbeit ist dem mathematischen Teilgebiet der *Optimierung* zuzuordnen, welches sich mit der Suche des Maximums bzw. Minimums einer Zielfunktion beschäftigt. In dieser Arbeit werde ich konkret anhand von Linearer Optimierung aufzeigen, dass eine fundierte Kenntnis von linearer Ungleichungs- und Gleichungssysteme sehr praxisorientiert eingesetzt werden kann. Lineare Optimierung ist ein Subbereich der Optimierung, der die Maximierung bzw. Minimierung einer linearen Zielfunktion unter linearen Nebenbedingungen beschreibt. Ich habe bewusst den Fokus auf Optimierungsprobleme in zwei Variablen zugunsten der Verständlichkeit gesetzt, denn in solchen graphisch lösbaren Problemen kommen fast alle Eigenschaften Linearer Programme zum Ausdruck. Einleitend bespreche ich die mathematischen Grundlagen Linearer Optimierung. Der Hauptteil dieser Arbeit thematisiert die praktische Anwendung von einfachen Beispielen in unserem Alltag, anhand derer der Prozess Lineare Optimierung erklärt werden sollte. Der Schluss zeigt eine rechnerische Methode zur Lösung von Linearen Programmen auf. Ziel meiner Arbeit ist es, auf einfache Weise zu vermitteln, wie Mathematik in der Wirtschaft eingesetzt werden kann. Meine Arbeit basiert größtenteils auf Sekundärliteratur. Ein Teil der methodischen Vorgehensweise jedoch besteht aus einem Interview mit DDr. Philipp Hungerländer, den ich über seine Arbeit als Optimierer befragte.

Vorwort

Die Freude an der Mathematik und ihrer formal exakten Sprache waren für mich Gründe dafür, meine VWA im Bereich Mathematik zu schreiben. Angestoßen von meiner Mathematikprofessorin und Betreuungsperson fand ich Zugang zu dem Thema mathematische Optimierung, wobei ich beschloss, mich auf Optimierung von linearen Funktionen zu konzentrieren.
Nach längerer Beschäftigung mit Fachliteratur und einer Auseinandersetzung mit dem Definition-Satz-Beweis-Schema im Zuge meines IT-Ferialpraktikums an der Alpen-Adria-Universität Klagenfurt im Bereich *Angewandte Mathematik* hat sich für mich herausgestellt, dass die theoretischen Aspekte der Linearen Optimierung und die logisch abstrakten Zusammenhänge innerhalb der Linearen Algebra für mich von großem Interesse sind, weswegen im Anhang auch mathematische Grundlagen besprochen werden.
Meine große Wissbegier an diesem Thema und speziell an den rechnerischen Methoden für Probleme in mehreren Variablen waren Grund dafür, diese Arbeit, eine längere Version der verpflichtenden VWA, zu verfassen, die sich größtenteils mit dem Simplex-Verfahren beschäftigt.
Trotz zahlreichen Überprüfens von Tipp- und Flüchtigkeitsfehler ist es sehr gut möglich, dass mir Fehler unterlaufen sind. Ich wäre Ihnen deswegen dankbar für eine Mitteilung, aber auch für konstruktive Kritik und Verbesserungsvorschläge, um diese Arbeit laufend aktualisieren zu können.

E-Mail-Adresse: tripp.markus@icloud.com

Spittal/Drau, am 29. Dezember 2018, Markus Tripp

Inhaltsverzeichnis

LaTeX-Schriftsatz

1

Einleitung

„Man soll die Dinge so einfach machen wie möglich - aber nicht einfacher.“

Albert Einstein (Zit. n. Brünner, o.J.)

Dieses Zitat von Albert Einstein war der Leitgrundsatz meiner Arbeit. Mein Anliegen ist es, der Leser*innenschaft Mathematik auf möglichst einfache Art und Weise näherzubringen. Daher habe ich mich bewusst auf Optimierungsprobleme in zwei Variablen, deren Vorteil in der graphischen Lösbarkeit liegt, konzentriert. Die Arbeit spannt einen Bogen von den theoretischen Grundlagen bis hin zu praktischen Anwendungen in unserem Alltag. In den einzelnen Kapiteln werde ich zunächst die mathematische Basis für Lineare Programme beschreiben, um mich dann auf Grundbegriffe und Eigenschaften Linearer Optimierung beziehen zu können. Abschließend werde ich anhand eines einfachen Beispiels einen vollständigen Planungsprozess eines Linearen Programms aufzeigen. Der zusätzliche Teil, der im Zuge einer weiterführenden Version der offiziellen VWA entstanden ist, beschäftigt sich mit einem rechnerischen Lösungsverfahren. Der größtenteils theoretische Anhang A zu den mathematischen Grundlagen ist dem Umstand der Eleganz des mathematischen Konstrukts und der Zusammenhänge in der Linearen Algebra, die mich zutiefst fasziniert haben, geschuldet. Er ist essentiell, um die späteren Linearen Programme in mehreren Variablen zu verstehen, denn die als Nebenbedingungen vorkommenden linearen Ungleichungssysteme werden als unterbestimmte, lineare Gleichungssysteme umgeschrieben, um sie rein algebraisch lösen zu können. (vgl. Scheid/Schwarz, 2009, S. 156) Zusätzlich ist die Matrizenschreibweise von enormer Wichtigkeit, um später Lineare Programme übersichtlich und vereinfacht darzustellen. Im Anhang B ist noch der erwähnte Algorithmus zur Lösung von linearen Gleichungssysteme als Flowchart und weiterführend als C++-Code dargestellt.

Im Zuge meines IT-Ferialpraktikums hatte ich auch die Möglichkeit mit Herrn DDr. Philipp Hungerländer, dessen Spezialgebiet die mathematische Optimierung ist, ein Interview zu führen, um mein Wissen in diesem Bereich zu vertiefen und praktische Anwendung in der realen Welt kennen zu lernen. Zurzeit arbeitet er an einer Optimierung für Rail Cargo mit dem Ziel, un-

ter bestimmten Nebenbedingungen möglichst wenig Lokomotiven zu verwenden, um alle Fahrten der Züge durchzuführen. Jedoch kommen hierbei mehrere Arten von mathematischer Optimierung zum Einsatz. Grundsätzlich handelt es sich um Ganzzahlige Optimierung, bei der zusätzlich zu den linearen Nebenbedingungen und Nichtnegativitätsbedingungen noch Ganzzahligkeitsbedingungen für die Variablen gelten. Hierbei kommen Lineare Programme als Subroutine vor. Abschließend erklärte er die Unterschiede der Lösungsverfahren Simplex- und Innere-Punkte-Methode, die sich größtenteils in der Laufzeit und Restartfähigkeit bemerkbar machen, sowie die Nutzung des Servers für die Lösung von Optimierungsproblemen. (vgl. Hungerländer, persönliche Kommunikation, 23. Juli, 2018; siehe Anhang C)
Mein Ziel ist es, der Leser*innenschaft zu zeigen, dass eine fundierte Kenntnis von *linearen Ungleichungen und Gleichungen* sehr nützlich ist und praxisorientiert eingesetzt werden kann.

2

Einführung

„Die Mathematik ist das Instrument, welches die Vermittlung bewirkt zwischen Theorie und Praxis, zwischen Denken und Beobachten: [...] Daher kommt es, daß [sic!] unsere ganze gegenwärtige Kultur, soweit sie auf der geistigen Durchdringung und Dienstbarmachung der Natur beruht, ihre Grundlage in der Mathematik findet.“

David Hilbert (Zit. n. Brünner, o.J.)

Die *„Vermittlung zwischen Theorie und Praxis“* findet zum Beispiel Ausdruck bei den Methoden der Linearen Optimierung. Diese Anwendung macht es möglich, zahlreiche Problemstellungen zu lösen, deren Ziel eine Minimierung bzw. Maximierung ist. Der erste Gedanke zu Optimierungsprozessen führt vermutlich zu ökonomische Probleme, wie die Maximierung des Gewinns oder die Minimierung der Kosten. Doch heutzutage finden solche Praktiken vor allem auch im Transportwesen und im technischen Bereich, wie bei der optimalen Ventilsteuerung von Verbrennungsmotoren, Anwendung. Gerade im Zeitalter der Digitalisierung und der Verdrängung vieler Arbeitsbereiche ist das Interesse an Optimierungsverfahren groß. (vgl. Koop/Moock, 2008, S. VII ff)

2.1 Mathematische Hintergründe

Bereits in der 11. Schulstufe kommen Schüler*innen im Mathematikunterricht mit Extremwertaufgaben in Kontakt, die im Zuge der Differentialrechnung behandelt werden. Doch beschränkt man sich bei solchen schulischen Beispielen auf eine Nebenbedingung, die die zu optimierende Funktion einschränkt. Im Gegensatz dazu ist es mit einer fundierten Kenntnis von *linearen Ungleichungssystemen* bereits möglich, praktische Optimierungsanwendungen zu lösen, die auch in der Wirtschaft und anderen Bereichen des Lebens Anwendung finden. (vgl. Schneider, o.J.a)

2.1.1 Lineare Ungleichungen

Definition 1. Als lineare Ungleichung in zwei Variablen in Normalform versteht man eine Ungleichung der Form $a_1x_1 + a_2x_2 < b$ ($\leq, \geq, >, \neq$) mit $a_1, a_2 \in \mathbb{R} \setminus \{0\}, b \in \mathbb{R}$. (vgl. Blaha, o.J., S. 105)

Um den zulässigen Bereich der Ungleichung zu finden, der jene Werte beinhaltet, die die Ungleichung erfüllen, kann es hilfreich sein, die Ungleichung in die allgemeine Form $x_2 < a_1'x_1 + b'$ überzuführen. Da die mathematische Interpretation der Gleichung $x_2 = a_1'x_1 + b'$ eine Gerade im Koordinatensystem ergibt und zusätzlich aber gilt, dass x_2 kleiner sein muss als der rechte Term, führt dies zu der Erkenntnis, dass die Lösungsmenge unterhalb der Geraden liegen muss. Die lineare Ungleichung ergibt somit eine *Halbebene*, die durch eine *Randgerade* begrenzt wird. (vgl. Blaha, o.J., S. 105)
Ähnlich wird bei Ungleichungen, in denen andere Relationszeichen auftreten, vorgegangen. Hierbei wird lediglich zwischen den Fällen unterschieden, ob die Halbebene über oder unter der Randgeraden liegt und ob die Randgerade Element der Lösungsmenge ist oder nicht. (vgl. Blaha, o.J., S. 105)

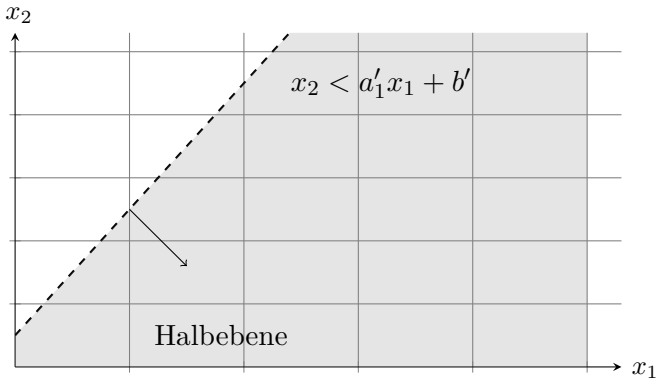

Abb. 1: Halbebene der Ungleichung $x_2 < a_1'x_1 + b'$
(eigene Darstellung in Anlehnung an vgl. Blaha, o.J., S. 105)

2.1.2 Lineare Ungleichungssysteme

Lineare Ungleichungssysteme bestehen wie LGS[1] aus mehreren Komponenten, in diesem Fall Ungleichungen, deren Korrektheit gleichzeitig erfüllt werden muss. Zweckmäßigerweise wird zur Auffindung der Lösung, die als *Durchschnitt der Halbebenen* zu interpretieren ist, die graphische Methode verwendet. Normalerweise ergibt sich aus dem Durchschnitt ein Bereich von Werten, die die Ungleichungen erfüllen. (vgl. Blaha, o.J., S. 107)

[1] LGS: lineares Gleichungssystem

Gegeben sei folgendes *lineares Ungleichungssystem:*

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 \leq b_1 \\
&a_{21}x_1 + a_{22}x_2 \leq b_2 \\
&\vdots \\
&a_{m1}x_1 + a_{m2}x_2 \leq b_m \\
&x_1 \geq 0, x_2 \geq 0.
\end{aligned}
$$

Die Menge M der Zahlenpaare (x_1, x_2), die die Ungleichungen erfüllen, bezeichnet man als *Menge der zulässigen Punkte.* Zur Lösung dieses System sind noch einige Gedankenschritte erforderlich:

- Wir wir bereits wissen, ergibt die Gesamtheit der Werte, die die Ungleichung $a_{i1}x_1+a_{i2}x_2 \leq b_i$ mit $|a_{i1}| + |a_{i2}| > 0$ erfüllen, eine Halbebene, die von der Randgerade $a_{i1}x_1 + a_{i2}x_2 = b_i$ begrenzt wird. (vgl. Burkard, o.J., S. 13)

- Die Lösung eines Ungleichungssystem von zwei Ungleichungen ist als Durchschnitt zweier Halbebenen zu interpretieren. Diese Lösungsmenge kann leer oder eine konvexe polyedrische Menge sein, die durch endlich viele Geraden begrenzt wird. (vgl. Burkard, o.J., S. 13 f) Zusätzlich bezeichnen wir eine Menge, deren Elemente innerhalb von Schranken liegen, als beschränkt. Andernfalls nennen wir sie unbeschränkt (nach unten und/oder oben hin). (vgl. Stobitzer, o.J.)

- Zwischen diesen Fällen wird auch beim Durchschnitt von m Ungleichungen unterschieden. (vgl. Burkard, o.J., S. 14)

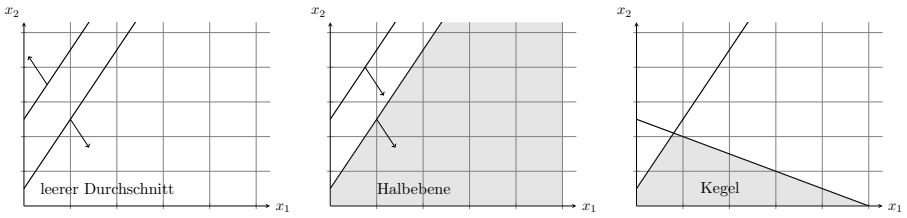

Abb. 2: Fälle für den Durchschnitt zweier Halbebenen
(eigene Darstellung in Anlehnung an Burkard, o.J., S. 14)

Was nun einzuwerfen ist, ist, dass wir uns bei der Lösung linearer Ungleichungssysteme auf die graphische Methode verlassen haben. Solche Darstellungen sind eine gute Methode - auch für die später auftretenden Optimierungsprobleme - um die wesentlichen Eigenschaften zu erkennen. Problematisch hierbei ist natürlich, dass man sich auf zwei Variablen beschränken muss, um graphisch an das Problem herangehen zu können. (vgl. Burkard, o.J., S. 13) Bei linearen Ungleichungssystemen mit mehreren Variablen werden sogenannte Schlupfvariablen eingeführt, die aus einem linearen Ungleichungssystem ein LGS machen und es so erlauben, rein algebraisch bei der Lösung vorzugehen. (vgl. Scheid/Schwarz, 2009, S. 156) Ein Verfahren zur Lösung eines LGS wird ausführlich im Anhang A.4.1 behandelt.

3

Lineare Optimierung

Ein grundlegender Vorteil Linearer Optimierung ist die vielseitige Einsatzmöglichkeit, die im folgenden Abschnitt näher beleuchtet wird.

3.1 Anwendungsgebiete Linearer Optimierung

Es existieren zahlreiche Gebiete, die mit den Methoden der Linearen Optimierung lösbar sind. Die wichtigsten Anwendungsgebiete sind:

- Produktionsplanungsprobleme
- Mischprobleme
- Investitionsplanungsprobleme
- Transportprobleme
- Zuordnungsprobleme
- Verschnittprobleme
- Knapsack-Probleme (vgl. Scheid/Schwarz, 2009, S. 4)

Im Abschnitt 3.2 werden relevante Grundbegriffe anhand eines Problems aus der Produktionsplanung geklärt, im Abschnitt 3.6 sind noch zusätzliche konkrete Beispiele und deren Modellierungen zu finden.

3.2 Grundbegriffe anhand eines einführenden Beispiels aus der Produktionsplanung

Beispiel 1 (Produktionsplanung)**.** *Ein agrarökonomischer Betrieb verfügt über* 40 *ha Anbaufläche,* 12 000, − *Kapital und dem Betrieb stehen* 240 *Arbeitstage zur Verfügung. Ziel ist es, jeweils Weizen und Zuckerrüben so anzubauen, dass unter Einhaltung der Restriktionen auf zur Verfügung stehende Anbaufläche, Kapital und Zeit der Erlös maximiert werden kann. (vgl. Scheid/Schwarz, 2009, S. 155)*

	Weizen	Zuckerrüben
Kosten/ha	200, −	600, −
Tage/ha	5	10
Erlös/ha	1000, −	1200, −

Tab. 1: Tabelle zu Beispiel 1 in Anlehnung an Scheid/Schwarz, 2009, S. 155

Die in der Tabelle angegebenen *Nebenbedingungen* beschreiben ein System von fünf linearen Ungleichungen, die die zu optimierenden Anzahlen beschränken,

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_1 + x_2 \leq 40, \quad 200x_1 + 600x_2 \leq 12\ 000, \quad 5x_1 + 10x_2 \leq 240,$$

wobei x_1, x_2 die Menge der Feldfrüchte in ha beschreibt. Die ersten beiden Ungleichungen werden als *Nichtnegativitätsbedingungen* bezeichnet, die die Lösung des Systems auf den ersten Quadranten beschränken, weil die Anzahl der Feldfrüchte, die angebaut werden, nicht negativ sein kann. (vgl. Koop/Moock, 2008, S. 14) Ihre Lösungsmenge wird durch ein Fünfeck im x_1x_2-Koordinatensystem, der Planungspolyeder[2] der Aufgabe, dargestellt, wobei es sich in diesem Fall bei der Verwendung von zwei Variabeln um ein Vieleck handelt. Diesen Planungsbereich erhält man durch die Schnittpunkte der Geraden, die die enstehenden Halbebenen der Ungleichungen beschränken. Die zu optimierende Funktion wird *Zielfunktion* genannt und ist in diesem Kontext durch den mehrgliedriger Term $z = F(x_1, x_2) = 1000x_1 + 1200x_2$ zur Berechnung des Verkaufserlöses gegeben. (vgl. Scheid/Schwarz, 2009, S. 155 f)
Nun ist derjenige Wert der Zielfunktion zu finden, der unter Einhaltung der Nebenbedingungen den optimalen Wert für z ergibt. Durch Parallelverschieben der Geradengleichung $1000x_1 + 1200x_2 = z$ entlang des Planungsvielecks ist der größtmögliche Wert der Zielfunktion abzulesen. Für den optimalen Wert wird der Rand des Planungsbereichs erwartet, der Punkt $(32, 8)$ (siehe dazu auch Abb. 3), welcher zu dem Funktionswert

$$\max\ 1000x_1 + 1200x_2 = 1000 \cdot 32 + 1200 \cdot 8 = 41\ 600$$

führt. Nun weiß der Betrieb, dass unter den gegebenen Bedingungen der höchstmögliche Gewinn zu erzielen ist, wenn 32 ha Weizen und 8 ha Zuckerrüben angebaut werden. (vgl. Scheid/Schwarz, 2009, S. 155 f)

[2]Polyeder: Vielflächner (vgl. Dudenredaktion, Polyeder)

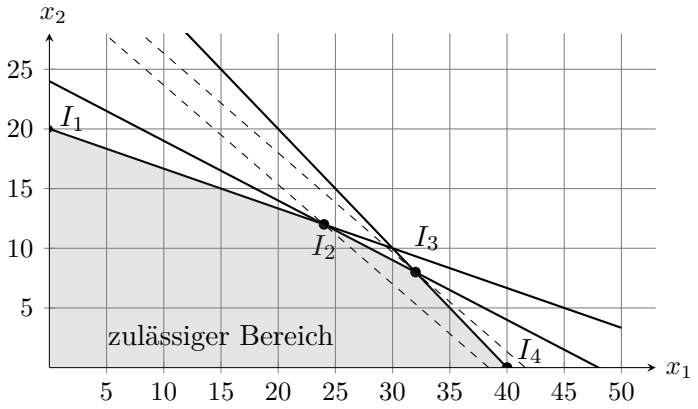

Abb. 3: Planungsvieleck und Zielfunktion des Beispiels 1
(eigene Darstellung in Anlehnung an Scheid/Schwarz, 2009, S. 155)

Formal geschrieben ergibt sich aus den Zusammenhängen folgendes mathematisches Modell, welches die Nebenbedingungen, die Restriktionen auf die Zielfunktion, und die Zielfunktion beschreibt. Im Abschnitt 3.6 werden noch weitere Bespiele zu Optimierungsmodellen beleuchtet.
Mathematische Modellierung:

$$x_1, x_2 \quad \text{Menge der Feldfrüchte in ha}$$

$$\max z = F(x_1, x_2) = 1000x_1 + 1200x_2$$

$$\begin{aligned} \textbf{u.d.N.}\ ^{3}\ x_1 + x_2 &\leq 40 \text{ (Beschränkung Anbaufläche)} \\ 200x_1 + 600x_2 &\leq 12\,000 \text{ (Beschränkung Kosten)} \\ 5x_1 + 10x_2 &\leq 240 \text{ (Beschränkung Tage)} \\ x_1, x_2 &\geq 0 \text{ (Anbau nur nichtnegativer Mengen)} \end{aligned}$$

Allgemein lässt sich über die Optimierung sagen, dass der optimale Punkt, für den eine Zielfunktion maximal bzw. minimal wird, am Rand des Planungsvielecks liegt. (vgl. Schwarz, o.J., S. 7)
Dieses einführende Beispiel sollte Aufschluss darüber geben, wie Lineare Optimierung in Grundzügen funktioniert und wie man bei der Lösung vorgeht. Die Eigenschaften eines solchen Problems sind am besten bei zwei Variablen ersichtlich, da es graphisch lösbar ist. Es ist jedoch der Fall, dass bei den meisten realen Optimierungsproblemen mehrere Variablen auftreten, weswegen man auf andere Lösungsverfahren zurückgreifen muss. (vgl. Burkard, o.J., S. 13)

[3] u.d.N.: unter den Nebenbedingungen (vgl. Koop/Moock, 2008, S. 41)

3.3 Mathematische Formulierung eines allgemeinen LP in zwei Variablen

> Unter einem *linearen Optimierungsproblem (LOP)* bzw. einem *Linearen Programm (LP)* ist eine Extremwertaufgabe mit Nebenbedingungen zu verstehen, wobei sowohl die zu maximierende bzw. minimierende Zielfunktion als auch die Nebenbedingungen von linearer Gestalt sind. Von allen zulässigen Lösungen ist also der Wert zu bestimmen, für den die Zielfunktion optimal wird. (vgl. Seiffart/Manteuffel, 1974, S. 10)

Gegeben sei folgendes *Lineare Programm* in zwei Variablen:

Zielfunktion Die zu optimierende Funktion heißt Zielfunktion.	$\max/\min\ z = F(x_1, x_2) = c_1x_1 + c_2x_2$
Nebenbedingungen (Restriktionen) Die Nebenbedingungen schränken die Zielfunktion ein.	$a_{11}x_1 + a_{12}x_2 \leq b_1$ $a_{21}x_1 + a_{22}x_2 \leq b_2$ $\vdots$ $a_{m1}x_1 + a_{m2}x_2 \leq b_m$
Nichtnegativitätsbedingungen	$x_1 \geq 0, x_2 \geq 0$ (vgl. Schneider, o.J.a)

Dabei sind

$z = F(x_1, x_2)$	Zielfunktion
x_1, x_2	Variable
c_1, c_2	reelle Koeffizienten der Zielfunktion
m	Anzahl der Nebenbedingungen
a_{ij}	reelle Koeffizienten der Nebenbedingungen mit $j = 1, 2$ und $i = 1, ..., m$
b_i	rechte Seite der Nebenbedingungen mit $i = 1, ..., m$ (vgl. Koop/Moock, 2008, S. 14)

3.4 Bedeutung des Begriffs Operations Research

Die Methoden der Linearen Optimierung sind einer der wichtigsten Bestandteile des Bereichs *Operations Research*. (vgl. Koop/Moock, 2008, S. 35)
Die ursprüngliche Entstehung des Begriffs *Operations Research* (OR) geht auf den militärischen Bereich während des Zweiten Weltkriegs zurück. Damit wurde die Entwicklung diverser Lösungsmethoden strategischer Probleme benannt. Nach Ende des Krieges prägte die Verwendung des Begriffs vor allem die Benutzung in der Wirtschaft. (vgl. Koop/Moock, 2008, S. 1)
Obwohl für die Bezeichnung heutzutage keine universell gültige Definition existiert, ist man sich größtenteils einig, dass *Operations Research* sich mit der *„Entwicklung und Anwendung mathematischer Methoden zur Entscheidungsvorbereitung“* beschäftigt. (Koop/Moock, 2008, S. 1)
Gegenwärtig wird unter dem vorliegenden Begriff eine Vielzahl mathematischer Methoden verstanden - Lineare Optimierung, Nichtlineare Optimierung und Graphentheorie - um einige zu nennen. (vgl. Koop/Moock, 2008, S. 2) Einen äußerst wichtigen Teil nehmen dabei die Methoden der Linearen Optimierung ein. Der größte Vorteil, den solche Systeme gegenüber anderen Lösungsinstrumenten aufweisen, ist, dass sie mathematisch einfach und übersichtlich dargestellt werden können. G.B. Dantzig entwickelte 1947 einen Algorithmus zur Lösung linearer Optimierungsprobleme, der unter dem Namen *Simplex-Verfahren* bekannt ist. Diese Lösungsmethode bildet gegenwärtig das klassische Instrument zur Lösungsermittlung. (vgl. Seiffart/Manteuffel, 1974, S. 9)

3.5 OR-gestützter Planungsprozess

„Das Problem zu erkennen ist wichtiger als die Lösung zu erkennen, denn die genaue Darstellung des Problems führt zur Lösung.“

Albert Einstein (Zit. n. Sonne/Weiß, 2013, S. 177)

Auch bei Linearer Programmierung ist die Modellierung des Problems zentral und stellt oft das eigentliche Problem der Optimierung dar, weil es aufgrund von zahlreichen Anwendungsgebieten kaum eine systematische Lösung bzw. Herangehensweise zur Darstellung des Prozesses gibt.
Der Anfang von jedem Optimierungsproblem beginnt beim Erkennen des Problems und der schrittweisen zunächst verbalen Beschreibung und Analyse. Im Allgemeinen werden restriktiv wirkende Daten (Nebenbedingungen) und Zielgrößen (Zielfunktion) gesucht und interpretiert. Die verbale Beschreibung geht mit der mathematischen Modellierung einher. (vgl. Koop/Moock, 2008, S. 2 f) Laut Definition von Martin Burger, versteht man unter einem mathematischen Modell eine berechenbare (vereinfachte) Menge von mathematischen Vorschriften, Gleichungen und Ungleichungen, die der Abbildung eines realen Vorgangs dient. Dabei ist es grundlegend, zu verstehen, dass es sich immer um eine im mathematischen Sinne Simplifizierung der Problemstellung handelt, weil eine exakt getreue Darstellung unmöglich scheint. (vgl. Burger, 2006, S. 3) Auf Grundlage des erhobenen mathematischen Systems sucht man nach einer Lösung des Problems, die darauffolgend auf die Durchführbarkeit untersucht und hinsichtlich der Nützlichkeit evaluiert wird. (vgl. Koop/Moock, 2008, S. 2 f)

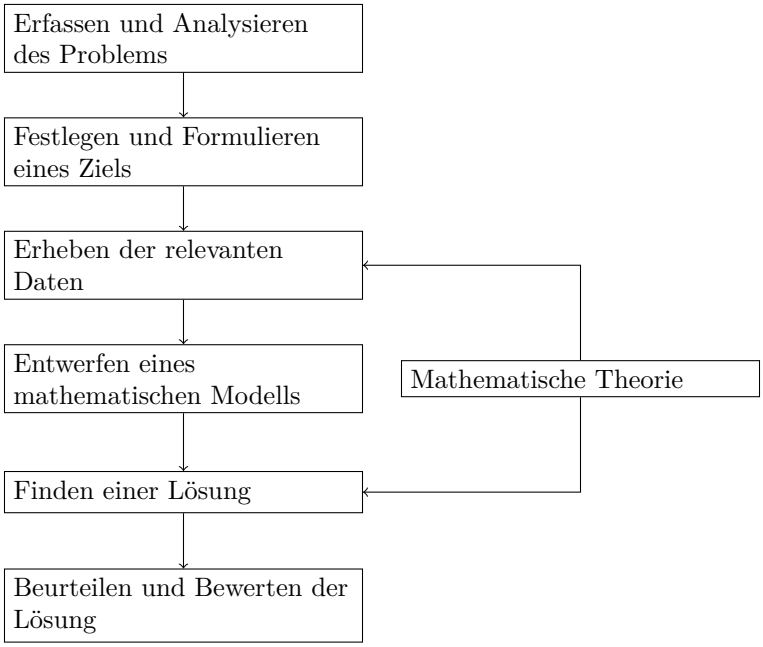

Abb. 4: OR-gestützter Planungsprozess
(eigene Darstellung in Anlehnung an Koop/Moock, 2008, S. 3)

Im Groben basiert ein vollständiger OR-gestützter Planungsprozess auf den in Abb. 4 dargestellten sechs Schritten.

3.6 Darstellung linearer Modelle anhand von ausgewählten Beispielen

Der Vorteil Linearer Optimierung liegt de facto unter anderem in der relativ einfachen Erstellung von Optimierungsmodellen. Es ist nicht nur möglich komplexe Zusammenhänge mathematisch übersichtlich darzustellen, sondern auch die Beschreibung von auf den ersten Blick nichtlinearen Problemen tut dem keinen Abbruch.(vgl. Seiffart/Manteuffel, 1974, S. 9)

Beispiel 2 (Produktionsplanung)**.** *Ein pharmazeutischer Betrieb stellt drei medizinische Präparate zur Bekämpfung einer Epidemie her, die jedoch alle etwa gleich viel Penicillin enthalten. Es bestehen allerdings Beschränkungen, die in der untentstehenden Tabelle ersichtlich sind. Wie sieht die optimale Verteilung aus, wenn eine möglichst große Gesamtmenge erwünscht ist? (vgl. Kronfellner/Peschek, 1986, S. 201)*

Abteilung	Herstellungszeit			Kapazität
	P_1	P_2	P_3	
1	-	4	10	300
2	10	3	-	500
3	4	5	2	350

Tab. 2: Tabelle zu Beispiel 2 in Anlehnung an Kronfellner/Peschek, 1986, S. 201

Mathematische Modellierung:

$$x_{P1}, x_{P2}, x_{P3} \quad \text{Anzahl der produzierten Präparate}$$

$$\max z = F(x_{P1}, x_{P2}, x_{P3}) = x_{P1} + x_{P2} + x_{P3}$$

$$\begin{aligned} \textbf{u.d.N.}\ 4x_{P2} + 10x_{P3} &\leq 300 \text{ (Kapazität Abteilung 1)} \\ 10x_{P1} + 3x_{P2} &\leq 500 \text{ (Kapazität Abteilung 2)} \\ 4x_{P1} + 5x_{P2} + 2x_{P3} &\leq 350 \text{ (Kapazität Abteilung 3)} \\ x_{P1}, x_{P2}, x_{P3} &\geq 0 \text{ (Produktion nur nichtnegativer Mengen)} \end{aligned}$$

Langwierige Berechnungen mit Hilfe des Simplex-Verfahrens ergeben die Optimallösung:

$$x_{P1} = 41, x_{P2} = 30, x_{P3} = 18.$$

Beispiel 3 (Ganzzahlige Optimierung)**.** *Ein Unternehmen, das sich auf die Herstellung von Leisten spezialisiert hat, will die Stückzahl durch optimale Verteilung erhöhen. Zur Verfügung stehen dem Unternehmen je 300 Leisten zu einer Länge von 6,5 m und 80 Leisten zu einer Länge von 5,5 m. Jedoch werden Leisten mit einer Länge von 2 m und Leisten mit einer Länge von 1,5 m gebraucht. Beachtet werden muss, dass doppelt so viele Leisten mit einer Länge von 1,5 m gebraucht werden. (vgl. Seiffart/Manteuffel, 1974, S. 20)*

Leiste zu 6,5m	Anzahl der 2-m-Leisten	Anzahl der 1,5-m-Leisten
1. Variante	3	0
2. Variante	2	1
3. Variante	1	3
4. Variante	0	4
Leiste zu 5,5m		
5. Variante	2	1
6. Variante	1	2
7. Variante	0	3

Tab. 3: Tabelle zu Beispiel 3 in Anlehnung an Seiffart/Manteuffel, 1974, S. 20 f

Mathematische Modellierung:

$x_1, x_2, x_3, x_4, x_5, x_6, x_7$ Anzahl der Anwendungen der sieben Varianten

$$\max z = F(x_1, x_2, x_3, x_5, x_6) = 3x_1 + 2x_2 + x_3 + 2x_5 + x_6$$

$$\begin{aligned} \textbf{u.d.N.}\ x_1 + x_2 + x_3 + x_4 &= 300 \text{ (Beschränkung der Leisten zu 6,5 m)} \\ x_5 + x_6 + x_7 &= 80 \text{ (Beschränkung der Leisten zu 5,5 m)} \\ 2\,(3x_1 + 2x_2 + x_3 + 2x_5 + x_6) - & \\ (x_2 + 3x_3 + 4x_4 + x_5 + 2x_6 + 3x_7) &= 0 \text{ (Beschränkung Leistenanzahl)} \\ x_1, x_2, x_3, x_4, x_5, x_6, x_7 &\in \mathbb{N}_0 \text{ (Verteilung nur nichtnegativer, ganzer Anzahlen)} \end{aligned}$$

Dieses Beispiel zur Darstellung von linearen Modellen stammt aus dem Bereich der Ganzzahligen Optimierung. Da ein solches Problem aber sehr komplexe Lösungsverfahren verlangt, würde die Erläuterung den Rahmen dieser VWA sprengen.

3.7 Mathematische Formulierung eines allgemeinen LP in n Variablen

Bereits in Abschnitt 3.3 haben wir das lineare Modell in zwei Variablen konkretisiert, nun wollen wir ein allgemeines Lineares Programm definieren.

Definition 2 (Lineares Optimierungsproblem)**.** „Unter einem *linearen Optimierungsproblem* bzw. einem *Linearen Programm (LP)* versteht man die Aufgabe, eine *lineare Zielfunktion* der Gestalt

$$z = F(x_1, ..., x_n) = \sum_{j=1}^{n} c_j x_j + d$$

zu maximieren oder zu minimieren unter den *linearen Nebenbedingungen* (Restriktionen)

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad oder \quad \sum_{j=1}^{n} a_{ij} x_j = b_i \quad oder \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i$$

für $i = 1, ..., m$ und unter den *Nichtnegativitätsbedingungen*

$$x_j \geq 0$$

für einige oder alle $j = 1, ..., n$.“ (Koop/Moock, 2008, S. 35 f)

3.8 Lösbarkeit eines LP

Obwohl in dieser Arbeit bis zu diesem Zeitpunkt nie von Vektoren die Rede war, möchte ich diese nun kurz einführen, ohne auf genaue Definitionen einzugehen. Ein LGS bildet einen Vektorraum, weil die n-Tupel eine algebraische Struktur aufweisen, die sie sich zum Beispiel bei der Vervielfachung bzw. bei der Addition von n-Tupeln bemerkbar macht. (vgl. Scheid/Schwarz, 2009, S. 12) Dieser Raum der Vektoren mit n reellen Komponenten wird mit R^n notiert, wobei in diesem Raum alle bekannten Rechenregeln für Skalare und Vektoren anwendbar sind. (vgl. Koop/Moock, 2008, S. 17)
Die Lösbarkeit Linearer Programme wird grundsätzlich in drei Fälle unterteilt:

1. Das Lineare Programm ist *unzulässig*, weil kein Vektor x existiert, der zulässig ist, d.h. es gibt keinen Vektor, der alle Nebenbedingungen erfüllt. Somit ist der dazugehörige Planungspolyeder leer.

2. Das Lineare Programm ist *unbeschränkt*, weil die Zielfunktion z mit Vektoren x, die alle Nebenbedingungen erfüllen, beliebig groß gemacht werden kann. Somit exisitiert keine endliche Optimallösung.

3. Das Lineare Programm besitzt mind. einen *optimalen Wert* x, weil das Lineare Programm *beschränkt* ist und der Punkt x *zulässig* ist. (vgl. Hackl, o.J., S. 3)

Weiters ist es möglich, dass es unendlich viele, endliche Optimallösungen gibt. Das ist genau dann der Fall, wenn die Zielfunktion parallel zu einer Kante ist, auf der alle Punkte optimal sind. (vgl. Scheid/Schwarz, 2009, S. 156)

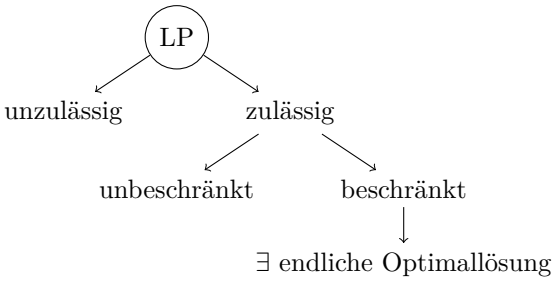

Abb. 5: Lösbarkeit eines LPs
(eigene Darstellung in Anlehnung an Hackl, o.J., S. 3)

In der unterstehenden Abbildung 6 wird die Lösbarkeit Linearer Programme anhand eines zweidimensionalen Maximierungsproblems graphisch aufgezeigt. Das linke Bild zeigt den Fall, dass kein zulässiger Vektor existiert, der Durchschnitt der Halbebenen, der Planungspolyeder dieser Aufgabe, ist somit leer. Das mittlere Bild beschreibt ein unbeschränktes Lineares Programm, weil die Zielfunktion beliebig groß gemacht werden kann. Das rechte Bild zeigt eine endliche Optimallösung.

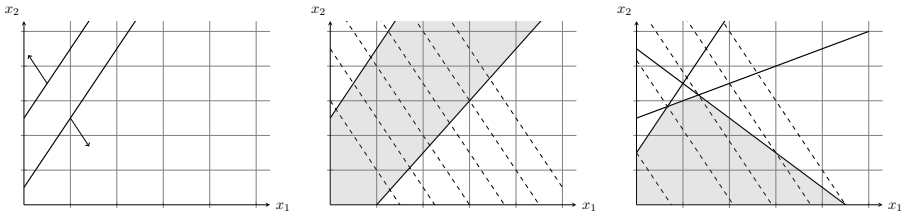

Abb. 6: Graphische Veranschaulichung der Lösungsfälle einer Maximierungsaufgabe
(eigene Darstellung)

3.9 Optimierungsprozess anhand eines Beispiels aus der Investitionsplanung

Im letzten Abschnitt dieser Arbeit wollen wir anhand eines Beispiels einen vollständigen OR-gestützten Optimierungsprozess aufzeigen, dessen sechs Schritte bereits im Abschnitt 3.5 näher beleuchtet wurden.

Erfassen und Analysieren des Problems

Beispiel 4 (Investitionsplanung)**.** *Eine Werbeagentur rät einer Partei, dass noch mindestens 30 einminütige und 20 dreiminütige Werbespots auf Youtube gebraucht werden, um Wählerstimmen in jugendlichen Kreisen markant zu erhöhen. Insgesamt sollen 150 Minuten an Werbung geschaltet werden. Der Konzern Google verkauft eine einminütige Werbezeit um* $450,-$*, eine*

dreiminütige Werbezeit um $1\ 000,-$. *(vgl. Kronfellner/Peschek, 1985, S. 150)*

Festlegen und Formulieren eines Ziels

Das *Ziel* ist es, mit minimalen Kosten dem Rat der Werbeagentur zu folgen und die Sympathie in jugendlichen Kreisen zu erhöhen, um Wählerstimmen zu fischen.

Erheben der relevanten Daten

Im ersten Schritt sollte man sich überlegen, welche Variablen eingeführt werden müssen und wofür diese stehen.

$$x_{1min}, x_{3min} \quad \text{Anzahl der ein- bzw. dreiminütigen Werbespots}$$

Die zu minimierende Funktion, die sogenannte *Zielfunktion*, beschreibt die Kosten der Werbespots.

$$z = F(x_{1min}, x_{3min}) = 450x_{1min} + 1000x_{3min} \quad \text{Kosten der Werbespots}$$

Die letzte Frage, die sich stellt, ist, welche restriktiven Größen auf die Zielfunktion wirken. Zunächst einmal gelten die *Nichtnegativitätsbedingungen* für die Variablen, weil keine negative Anzahl an Werbespots geschaltet werden kann. Wir beschränken uns also auf den ersten Quadranten.

$$x_{1min}, x_{3min} \geq 0 \quad \text{Schaltung nur nichtnegativer Anzahl an Werbespots}$$

Zusätzlich ergeben sich folgende Nebenbedingungen aus dem Kontext des Beispiels.

$$\begin{aligned} x_{1min} + 3x_{3min} &\geq 150 \quad \text{Schaltung von mindestens 150 min Werbung} \\ x_{1min} &\geq 30 \quad \text{Schaltung von mindestens 30 einminütiger Werbespots} \\ x_{3min} &\geq 20 \quad \text{Schaltung von mindestens 20 dreiminütiger Werbespots} \end{aligned}$$

Weiters ist dieses Lineare Programm ein Problem aus der Ganzzahligen Optimierung, da die Variablen nur natürliche Werte annehmen dürfen, wobei wir dies bei der Lösung nicht weiters beachten.

$$x_{1min}, x_{3min} \in \mathbb{N}_0 \quad \text{Schaltung nur nichtnegativer, ganzer Anzahlen an Werbespots}$$

Entwerfen eines mathematischen Modells

Zusammenfassend ergibt sich folgende mathematische Formulierung des vorliegenden Optimierungproblems:

Mathematische Modellierung:

$$x_{1min}, x_{3min} \quad \text{Anzahl der ein- bzw. dreiminütigen Werbespots}$$

$$\min z = F(x_{1min}, x_{3min}) = 450x_{1min} + 1000x_{3min} \text{ (Kosten der Werbespots)}$$

$$\begin{aligned}
\textbf{u.d.N. } & x_{1min} + 3x_{3min} \geq 150 \quad \text{(Schaltung von mindestens 150 min Werbung)} \\
& x_{1min} \geq 30 \text{ (Schaltung von mindestens 30 einminütiger Werbespots)} \\
& x_{3min} \geq 20 \text{ (Schaltung von mindestens 20 dreiminütiger Werbespots)} \\
& x_{1min}, x_{3min} \in \mathbb{N}_0 \text{ (Schaltung nur nichtnegativer, ganzer Anzahlen an Werbespots)}
\end{aligned}$$

Finden einer Lösung

Gesucht sind nun die Werte für x_{1min} und x_{3min}, die die Zielfunktion unter Beachtung der Nebenbedingungen minimieren. Das Lineare Programm lässt sich graphisch darstellen und lösen:

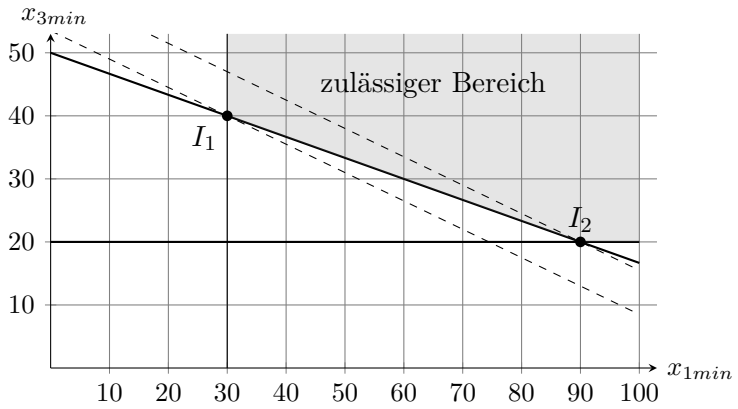

Abb. 7: Planungsvieleck und Zielfunktion des Beispiels 4
(eigene Darstellung)

Die Lösung des Ungleichungssystems entspricht der grauen Fläche, dem *Planungsbereich* dieses Beispiels, der durch den Durchschnitt der Halbebenen entstanden ist. Um die Lösung zu erhalten, verschiebt man nun die Geradengleichung $x_{3min} = -\frac{9}{20}x_{1min} + \frac{z}{20}$ parallel, die mindestens einen Punkt mit dem Planungsbereich gemeinsam hat, bis der Wert für z minimal wird. Der optimale Wert für die Zielfunktion ist am Rand des Planungsbereichs anzunehmen und beträgt

$$\min 450x_{1min} + 1000x_{3min} = 450 \cdot 30 + 1000 \cdot 40 = 53\,500.$$

Beurteilen und Bewerten der Lösung

Es existieren Werte, die alle Nebenbedingungen erfüllen, und somit *zulässig* sind. Zusätzlich ist der Planungsbereich nach oben hin *unbeschränkt*, was bei einer Maximierungsaufgabe in diesem Kontext zur Folge hätte, dass es keinen endlichen, optimalen Wert gibt.
Da es sich aber um eine Minimierungsaufgabe handelt, weiß die Partei, dass sie 30 einminütige und 40 dreiminütige Werbespots schalten muss, um dem Rat der Agentur zu folgen. Ihre Kosten belaufen sich auf $53\ 500,-$, die das Minimum unter den Anforderungen darstellen und somit optimal sind.

4

Simplex-Methode

Die Idee des Verfahrens, mit dem anders als bei der graphischen Herangehensweise auch Lineare Programme in mehreren Variablen gelöst werden können, ist, sich entlang der Eckpunkten des Zulässigkeitsbereichs zu bewegen, die kontinuierlich die Zielfunktion erhöhen. (vgl. Koop/Moock, 2008, S. 56) Aber bevor wir uns das Verfahren genauer ansehen, müssen wir nun eine Reihe von Begriffen einführen.

4.1 Normalform eines Linearen Programms

Unter der Normalform versteht man eine einheitliche Form eines Linearen Optimierungsproblems, die dazu dient, den Lösungsalgorithmus generalisieren zu können.

Definition 3 (Normalform eines LPs)**.** Die Normalform eines Linearen Programms hat die Gestalt:

$$\max\ z = F(x_1, ..., x_n) = \sum_{j=1}^{n} c_j x_j$$

$$\textbf{u.d.N.}\ \sum_{j=1}^{n} a_{ij} x_j = b_i \qquad \text{für } i = 1, ..., m,$$

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

Ein Lineares Programm in Normalform beschreibt also stets ein Maximierungsproblem. Ein Minimierungsproblem

$$\min\ Z = F(x_1, ..., x_n) = \sum_{j=1}^{n} c_j x_j$$

transformiert sich in Normalform in

$$\max\ z = -F(x_1, ..., x_n) = \sum_{j=1}^{n} -c_j x_j.$$

Durch die Einführung von sogenannten Schlupfvariablen lassen sich Ungleichungssysteme der

Form

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i$$

in lineare Gleichungssysteme der Gestalt

$$\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i$$

umschreiben. Analog wird aus

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i$$

ein Gleichungssystem der Form

$$\sum_{j=1}^{n} a_{ij}x_j - x_{n+i} = b_i.$$

Für die Schlupfvariablen gelten ebenfalls die Nichtnegativitätsbedingungen.

Durch die Definition der Matrizenschreibweise (siehe Anhang A.1) lässt sich das LP vereinfacht in der Form

$$\max z = F(x) = c^T x \tag{1}$$

$$\begin{aligned} \textbf{u.d.N.}\ & Ax = b \\ & x_j \geq 0 \qquad \forall j \end{aligned}$$

darstellen. (vgl. Koop/Moock, 2008, S. 40 ff)

Beispiel 5 (Transformation eines allgemeinen LP in Normalform)**.** *Gegeben ist das LP des Beispiels 1:*

$$\max z = F(x_1, x_2) = 1000x_1 + 1200x_2$$

$$\begin{aligned} \textbf{u.d.N.}\ & x_1 + x_2 \leq 40 \text{ (Beschränkung Anbaufläche)} \\ & 200x_1 + 600x_2 \leq 12\,000 \text{ (Beschränkung Kosten)} \\ & 5x_1 + 10x_2 \leq 240 \text{ (Beschränkung Tage)} \\ & x_1, x_2 \geq 0 \text{ (Anbau nur nichtnegativer Mengen)} \end{aligned}$$

Da es sich bereits um eine Maximierungsaufgabe handelt, bleibt die Zielfunktion gleich. Durch Einführung von Schlupfvariablen können wir die Ungleichungen bis auf die Nichtnegativitätsbedingung in Gleichungen umschreiben:

$$\begin{aligned} x_1 + x_2 \quad &+x_3 \quad &&= 40 \\ 200x_1 + 600x_2 \quad &+x_4 \quad &&= 12\ 000 \ \ (x_1, x_2, x_3, x_4, x_5 \geq 0) \\ 5x_1 + 10x_2 \quad &+x_5 \quad &&= 240. \end{aligned}$$

Dabei beschreibt x_3 die Anbaufläche, die nicht benutzt wurde. Analog gibt x_4 die finanziellen Mittel an, die nicht verwendet wurden, und x_5 die Zeit in Tagen, die nicht aufgewendet wurde. Nun haben wir folgendes Lineare Programm in Normalform erhalten:

$$\max z = F(x_1, x_2) = 1000x_1 + 1200x_2$$

$$\begin{aligned} \textbf{u.d.N.}\ x_1 + x_2 + x_3 &= 40 \\ 200x_1 + 600x_2 + x_4 &= 12\ 000 \\ 5x_1 + 10x_2 + x_5 &= 240 \\ x_1, x_2, x_3, x_4, x_5 &\geq 0. \end{aligned}$$

4.2 Instrument zur Lösungsfindung

Zusätzlich ist es noch sinnvoll zu definieren, dass für den Rang der Matrix $A \in \mathbb{R}^{m \times n}$ $r(A) = m < n$ gilt, denn

- gelte zum Beispiel $r(A) = m = n$, wäre die Lösung bereits eindeutig, was keinen Spielraum mehr lassen würde,
- wäre der Rang $r(A) < m$, dann wären die Restriktionen entweder redundant oder sogar widersprüchlich. (vgl. Koop/Moock, 2008, S. 45 f)

Bemerkung: Es gilt für den Rang einer Matrix $A \in \mathbb{R}^{m \times n}$ $r(A) \leq \min(m, n)$. (vgl. Koop/Moock, 2008, S. 22)

Satz 1. *Gegeben sei ein LP in Normalform, für das $r(A) = m < n$ gelte. Daraus folgt, dass es stets ein $x \in \mathbb{R}^n$ gibt, dass die Restriktionen $Ax = b$ erfüllt. (vgl. Koop/Moock, 2008, S. 45)*

Beweis 1. Da für die Matrix $r(A) = m$ gilt, ist es möglich m linear unabhängige Spalten der Matrix A auszuwählen. Diese Untermatrix von A, die wir mit $A_B \in \mathbb{R}^{m \times m}$ bezeichnen, besitzt den vollen Rang m, die restliche Matrix bezeichnen wir mit $A_N \in \mathbb{R}^{m \times (n-m)}$. Es gilt:

$$Ax = b \Leftrightarrow (A_B | A_N) \begin{pmatrix} x_B \\ x_N \end{pmatrix} = b \Leftrightarrow A_B x_B + A_N x_N = b.$$

Setzen wir nun $x_N = 0$, so ist x_B eindeutig bestimmt, denn die Untermatrix A hat den maximalen Rang m. (vgl. Koop/Moock, 2008, S. 45; vgl. Hackl, o.J., S. 3) (siehe auch A.4.2 und A.4.3) □

Beispiel 6 (Einführende Betrachtung von Basislösungen)**.** *Gegeben sind zwei Nebenbedingungen des Linearen Programms des Beispiels 1:*

$$200x_1 + 600x_2 \leq 12\ 000$$
$$5x_1 + 10x_2 \leq 240.$$

Zusätzlich gelten die Nichtnegativitätsbedingungen $x_1, x_2 \geq 0$. Weiters möchten wir ohne Angabe der Zielfunktion die Zielfunktion maximieren. Durch Einführung der Schlupfvariablen ergibt sich folgendes lineare Gleichungssystem

$$200x_1 + 600x_2 + x_3 = 12\ 000$$
$$5x_1 + 10x_2 + x_4 = 240.$$

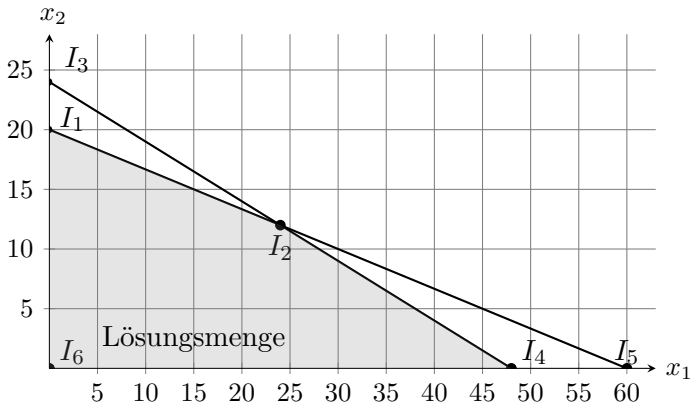

Abb. 8: Graphische Darstellung der Basislösungen des Beispiels 6
(eigene Darstellung)

Wir haben nun ein lineares Gleichungssystem mit $m = 2$ Gleichungen und $n = 4$ Variablen, das den Rang $r(A) = m < n$ besitzt, wodurch es m linear unabhänige Spaltenvektoren gibt, und wir möchten Eckpunkte des Zulässigkeitsbereich berechen, die die Zielfunktion maximieren. Wir wissen nämlich bereits, dass die Optimallösung an einer Ecke des Planungsbereichs anzunehmen ist. Man kann nur durch das Nullsetzen zweier Variablen eine Lösung berechnen, weil wir somit ein quadratisches System mit $m = n$ erhalten, das grundsätzlich genau eine Lösung besitzt. Graphisch gesehen wäre das in unserem Fall jeweils der Schnitt zweier Geraden, was auch die Abb. 8 symbolisiert, und leicht nachvollziehbar ist.

Aus dieser Überlegung heraus ergibt sich die Frage, wie viel verschiedene Möglichkeiten es gibt, von vier Variablen zwei null zu setzen. Mithilfe des Binomialkoeffizienten aus der Kombinatorik

ist es einfach zu sehen, dass es

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} = \frac{4 \cdot 3}{2 \cdot 1} = 6$$

verschiedene Möglichkeiten gibt, von sechs Variablen zwei auszuwählen und somit ergeben sich sechs potenzielle Lösungen. Die Variablen, die wir null setzen, bezeichen wir als *Nullvariablen* bzw. *Nichtbasisvariablen* und die Nichtnullvariablen nennen wir *Basisvariablen.*
Zur Lösung des Linearen Programms müssten wir also sechs Gleichungssysteme lösen und überprüfen, ob sie die Restriktionen erfüllen und die Zielfunktion maximieren. Die Lösungen dieser sechs Gleichungssysteme sind die eingezeichneten Punkte der Abb. 8, die Basislösungen der Aufgabe. Es fällt auf, welch immensen Rechenaufwand diese Herangehensweise bereits bei kleineren Problemen mit sich bringen würde, wenn man betrachtet, wie rasant der Binomialkoeffizient wächst, womit diese Methode zur Lösungsfindung schnell durch bessere Algorithmen ersetzt wurde. (vgl. Schneider, o.J.b)

4.2.1 Basislösung

Im folgenden wollen wir genau definieren, was eine Basislösung ist, denn sie ist für Lineare Programme von hoher Relevanz.

Definition 4 (Basislösung)**.** Gegeben sei ein Lineares Programm in der Form (1) mit der zusätzlichen Bedingung $r(A) = m < n$. Weiters sei A_B eine Untermatrix von A, wobei B eine Indexmenge beschreibt, die angibt, welche Spalten von A in A_B enthalten sind. A_N seien die restlichen Spalten, wobei N angibt, welche Spalten von A nicht in A_B enthalten sind.

- Eine Lösung $x \in \mathbb{R}^n$ der Gleichung $Ax = b$ wird als Basislösung bezeichnet, sofern $x_N = 0$ ist.
- Die Komponenten von x, die zu A_B gehören, nennt man Basisvariablen (BV) und deren Wert ist durch $x_N = 0$ eindeutig bestimmt.
- Eine Basislösung heißt entartet oder degeneriert, wenn eine Basisvariable einer Basislösung gleich null ist. (vgl. Hackl, o.J., S. 4)

Der folgende Fundamentalsatz legitimiert die Herangehensweise des Beispiels 6. Er sagt aus, dass wir uns bei der Lösung eines LPs nur auf die Basislösungen, die Eckpunkte des Zulässigkeitsbereich, beschränken müssen.

Satz 2 (Fundamentalsatz der Linearen Optimierung)**.** *Gegeben sei ein Lineares Programm in der Form (1) mit der zusätzlichen Bedingung $r(A) = m < n$. Das LP hat genau dann eine zulässige Lösung, wenn es auch eine zulässige Basislösung gibt. Weiters existiert genau dann eine optimale Lösung, wenn es auch eine optimale zulässige Basislösung gibt. (vgl. Koop/Moock, 2008, S. 46)*

Zusammenfassung

Die vorherige Definitionen der Basislösung und des Fundamentalsatzes der Linearen Optimierung scheinen auf den ersten Blick etwas kompliziert und unverständlich. Aber was sagen sie genau aus? Zulässige Basislösungen sind genau Eckpunkte des Zulässigkeitsbereich, wie es in der Abb. 8 gut erkennbar ist. Der Fundamentalsatz der Linearen Optimierung sagt nun aus, sobald dieser Bereich nicht leer ist, d.h. eine zulässige Lösung besitzt, muss es eine Ecke geben. Weiters sagt er aus, dass diese Ecke zu finden sei, sofern eine Optimallösung existiert. (vgl. Hackl, o.J., S. 4) Somit muss man sich bei dem Untersuchen von Linearen Programmen nur auf Basislösungen konzentrieren.

4.3 Simplex-Verfahren

In diesem Abschnitt betrachten wir das ältesteste und zugleich bekannteste Verfahren zur Lösung eines Linearen Programms, das nach dem Zweiten Weltkrieg von G.B. Dantzig entwickelt wurde: das sogenannte Simplex-Verfahren. Ausgehend von einer zulässigen Lösung wird entlang der Ecken nach einer besseren Lösung gesucht bis der Zielfunktionswert nicht mehr verbessert werden kann. (vgl. Kronfellner/Peschek, 1986, S. 189)
Bei der Aufgabe 1 würde man sich zum Beispiel vom Eckpunkt $(0, 0)$ zu $(0, 20)$ zu $(24, 12)$ zu $(32, 8)$ bewegen, was sich in der Abb. 9 nachvollziehen lässt.

4.3.1 Tableaus

Tableaus sind schematische Darstellungen, die als Veranschaulichung des Simplex-Verfahrens fungieren und zur Handrechnung verwendet werden. Ein Tableau eines Linearen Programms in der Form (1) sieht folgendermaßen aus:

BV	x_1	x_2	$\cdots$	x_n	z	b
x_{j_1}	a_{11}	a_{12}	$\cdots$	a_{1n}	0	b_1
x_{j_2}	a_{21}	a_{22}	$\cdots$	a_{2n}	0	b_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
x_{j_m}	a_{m1}	a_{m2}	$\cdots$	a_{mn}	0	b_m
z	$-c_1$	$-c_2$	$\cdots$	$-c_n$	1	Z.-Wert

Die in der linken Spalte angeführten Variablen $x_{j_1}, ..., x_{j_m}$ werden als Basisvariable bezeichnet. Die ersten m Zeilen beschreiben das lineare Gleichungssystem $Ax = b$, die letzte Zeile die Zielfunktion, die sogenannte Ergebniszeile. Sie entsteht, indem man z als zusätzliche Variable einführt und die Zielfunktion als Gleichung ansieht:

$$z = c_1x_1 + ... + c_nx_n \mid -z \mid \cdot(-1)$$
$$0 = -c_1x_1 + ... - c_nx_n + z.$$

Dies führt zu folgendem Optimierungsproblem hin, das äquivalent zu dem der Form 1 ist:

$$\max z$$

$$\begin{aligned} \textbf{u.d.N } & Ax = b \\ & -c^T x + z = 0 \\ & x_j \geq 0 \qquad \forall j. \end{aligned}$$

Das Gleichungssystem liegt in kanonischer Form vor, wenn es $m+1$ verschiedene Einheitsvektoren enthält, die eine Basis bilden. Ein Simplextableau in kanonischer Form sieht folgendermaßen aus, wobei wir hier der Einfachheit halber angenommen haben, dass die Basisvariablen gerade die letzten m Variablen sind, was bei einem Ungleichungssystem $Ax \leq b$ am Anfang die Schlupfvariablen sind (vgl. Koop/Moock, 2008, S. 56 f):

BV	x_1	$\cdots$	x_{n-m}	x_{n-m+1}	$\cdots$	x_n	z	b
x_{n-m+1}	$a_{1,1}$	$\cdots$	$a_{1,n-m}$	1	$\cdots$	0	0	b_1
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
x_n	$a_{m,1}$	$\cdots$	$a_{m,n-m}$	0	$\cdots$	1	0	b_m
z	$-c_1$	$\cdots$	$-c_{n-m}$	0	$\cdots$	0	1	Z.-Wert

4.4 Primal-Simplex-Verfahren anhand eines einführenden Beispiels

Im Folgenden versuchen wir anhand des Beispiels 1 aus der Produktionsplanung die Grundidee und die Vorgehensweise des Simplex-Algorithmus zu erläutern. Zur Wiederholung:

Beispiel 7 (Primal-Simplex-Verfahren)**.** *Ein agrarökonomischer Betrieb verfügt über* 40 *ha Anbaufläche,* 12 000,− *Kapital und dem Betrieb stehen* 240 *Arbeitstage zur Verfügung. Ziel ist es, jeweils Weizen und Zuckerrüben so anzubauen, dass unter Einhaltung der Restriktionen auf zur Verfügung stehenende Anbaufläche, Kapital und Zeit der Erlös maximiert werden kann. (vgl. Scheid/Schwarz, 2009, S. 155)*

Daraus ergibt sich folgende Modellierung des dazugehörigen Linearen Programms:

$$\max z = F(x_1, x_2) = 1000x_1 + 1200x_2$$

$$\begin{aligned} \textbf{u.d.N.} \; x_1 + x_2 &\leq 40 \text{ (Beschränkung Anbaufläche)} \\ 200x_1 + 600x_2 &\leq 12\,000 \text{ (Beschränkung Kosten)} \\ 5x_1 + 10x_2 &\leq 240 \text{ (Beschränkung Tage)} \\ x_1, x_2 &\geq 0 \text{ (Anbau nur nichtnegativer Mengen)} \end{aligned}$$

Unsere Vorüberlegungen führen uns zu folgendem Ausgangstableau, durch welches bereits eine Vielzahl von Informationen abgelesen werden können:

BV	x_1	x_2	x_3	x_4	x_5	z	b
x_3	1	1	1	0	0	0	40
x_4	200	600	0	1	0	0	12 000
x_5	5	10	0	0	1	0	240
z	-1000	-1200	0	0	0	1	0

Die Schlupfvariablen x_3, x_4, x_5 sind die Basisvariablen, ihr Wert ist durch die Nichtbasisvariablen $x_1, x_2 = 0$ eindeutig festgelegt und da das Tableau in kanonischer Form vorliegt, leicht ablesbar:

$$x_3 = 40, \quad x_4 = 12\ 000, \quad x_5 = 240.$$

Ökonomisch interpretiert bedeutet dies Produktionsstillstand, es werden 0 ha Feldfrüchte angebaut, womit keine finanziellen Mittel und Arbeitszeit verwendet werden müssen. Graphisch wäre dies der Ursprung im $x_1 - x_2$-Koordinatensystem. Weiters lässt sich sagen, dass durch Betrachtung der Spalten beim Anbau

- von einem ha Weizen logischerweise die mögliche Anbaufläche um 1 ha sinkt, das Kapital um 200 GE reduziert wird und 5 Arbeitstage aufgebraucht werden. Im Gegenzug kann der Erlös um 1000 GE erhöht werden.
- von einem ha Zuckerrüben ebenfalls die mögliche Anbaufläche um 1 ha sinkt, das Kapital um 600 GE reduziert wird und 10 Arbeitstage aufgebracht werden. Dafür erlangt der Betrieb 1200 GE Erlös.

1. Iterationsschritt

   i. Bestimmung der in die Basis eintretenden Variable

   Wir überlegen uns, welche der beiden Fleldfrüchte einen größeren Einfluss auf den Erlös hat, d.h. welche der beiden Nichtbasisvariablen x_1, x_2 die Zielfunktion stärker maximiert. Dazu suchen wir das Minimum

   $$\min\{c_1^{(0)}, c_2^{(0)}\} = -1200 = c_2^{(0)}$$

   unter den Koeffizienten der Nichtbasisvariablen der Zielfunktion. Die hochgestellte 0 beschreibt r, sie gibt die Anzahl der Veränderungen des Tableaus an. Somit ist $j = 2$ die Pivotspalte und x_2 die in die Basis eintretende Variable.

BV	x_1	x_2	x_3	x_4	x_5	z	b
x_3	1	1	1	0	0	0	40
x_4	200	600	0	1	0	0	12 000
x_5	5	10	0	0	1	0	240
z	-1000	-1200	0	0	0	1	0

ii. Bestimmung der aus der Basis zu eliminierenden Variable

Im ersten Teilschritt haben wir uns dazu entschlossen, ausschließlich Feldfrüchte des Typs x_2 anzubauen. Nun fragen wir uns, wie viel *ha* wir davon anbauen können, ohne die Restriktionen zu verletzen. Wir berechnen je Gleichung den höchstzulässigen Wert für x_2. Relevant ist für uns nur der kleinste Wert, denn er gibt nämlich an, ab wann das ganze System unzulässig wird. Wir suchen also:

$$\min\{\frac{b_1^{(0)}}{a_{12}^{(0)}}, \frac{b_2^{(0)}}{a_{22}^{(0)}}, \frac{b_3^{(0)}}{a_{32}^{(0)}}\} = 20 = \frac{b_2^{(0)}}{a_{22}^{(0)}}.$$

Dieser Wert muss größer als null sein, weil andernfalls die Nichtnegativitätsbedingung verletzt wird. Somit ist $i = 2$ die Pivotzeile und die Kreuzung der Pivotspalte und Pivotzeile das Pivotelement, in unserem Fall a_{22}. Aus der Basis eliminiert wird x_4.

BV	x_1	x_2	x_3	x_4	x_5	z	b
x_3	1	1	1	0	0	0	40
x_4	200	$\underline{600}$	0	1	0	0	12 000
x_5	5	10	0	0	1	0	240
z	-1000	-1200	0	0	0	1	0

iii. Basiswechsel

Im letzten Teilschritt müssen wir in der Spalte der zur Basisvariable werdenden Variable (hier: x_2) einen Einheitsvektor erzeugen, damit das $(r+1)$-te Tableau wieder kanonische Form besitzt und ein Variablenaustausch möglich ist. Dazu multiplizieren wir die Pivotzeile mit $\frac{1}{a_{22}^{(0)}}$, wobei $a_{22}^{(0)}$ das Pivotelement ist und erhalten somit $a_{22}^{(1)} = 1$. Danach addieren wir das $-a_{12}^{(0)}$-fache der Pivotzeile der ersten Zeile, das $-a_{32}^{(0)}$-fache der Pivotzeile der dritten Zeile und das $-c_2^{(0)}$-fache der Pivotzeile der Ergebniszeile. Dann erhalten wir folgendes Simplextableau:

BV	x_1	x_2	x_3	x_4	x_5	z	b
x_3	$\frac{2}{3}$	0	1	$-\frac{1}{600}$	0	0	20
$\underline{x_2}$	$\frac{1}{3}$	1	0	$\frac{1}{600}$	0	0	20
x_5	$\frac{5}{3}$	0	0	$\frac{1}{60}$	1	0	40
z	-600	0	0	2	0	1	24 000

Dieses neue, zweite Simplextableau veranschaulicht wieder viele Informationen. Die neuen Basisvariablen x_3, x_4, x_5 haben die Werte

$$x_3 = 20, \quad x_2 = 20, \quad x_5 = 40.$$

Ökonomisch interpretiert bedeutet dies, dass wir beim Anbau von $x_2 = 20$, die vollen finanziellen Möglichkeiten ausgeschöpft haben ($x_4 = 0$), jedoch stehen uns noch Arbeitszeit und Anbaufläche zur Verfügung. Aber haben wir nun also schon unsere

Optimallösung gefunden? Nein, denn die Zielfunktion weist noch negative Koeffizienten auf. Das bedeutet, dass, wenn wir zusätzliche Einheiten anbauen, der Erlös noch erhöht werden kann. Wir setzen nun das Simplex-Verfahren iterativ fort bis die Zielfunktion nur mehr nichtnegative Werte enthält.

2. Iterationsschritt

BV	x_1	x_2	x_3	x_4	x_5	z	b
x_3	0	0	1	$\frac{1}{200}$	$-\frac{2}{5}$	0	4
x_2	0	1	0	$\frac{1}{200}$	$-\frac{1}{5}$	0	12
$\underline{x_1}$	1	0	0	$-\frac{1}{100}$	$-\frac{3}{5}$	0	24
z	0	0	0	-4	360	1	38 400

3. Iterationsschritt

BV	x_1	x_2	x_3	x_4	x_5	z	b
$\underline{x_4}$	0	0	200	1	-80	0	800
x_2	0	1	-1	0	$\frac{19}{5}$	0	8
$\underline{x_1}$	1	0	2	0	$-\frac{1}{5}$	0	32
z	0	0	800	0	40	1	41 600

Nun haben wir die Optimallösung gefunden:

$$x_1 = 32, \quad x_2 = 8.$$

Beim Anbau von $x_1 = 32$ und $x_2 = 8$ macht der Betrieb einen Erlös von 41 600 GE. (vgl. Schneider, o.J.b)

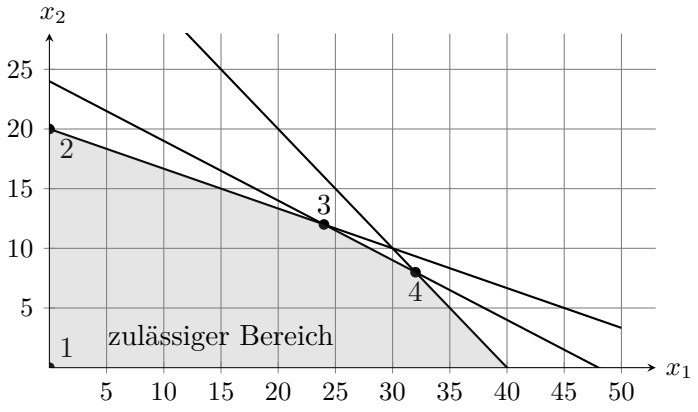

Abb. 9: Lösungsschritte des Beispiels 7
(eigene Darstellung)

4.5 Primal-Simplex-Verfahren

Das Primal-Simplex-Verfahren ist eine Methode zur Lösung von sogenannten *speziellen Maximumproblemen* der Form:

$$\max z = F(x) = c^T x \tag{2}$$

$$\textbf{u.d.N.} \sum_{j=1}^{n} a_{ij} x_j \leq b_i$$

$$x_j \geq 0$$

$$\text{mit } b_i \geq 0 \qquad \forall i.$$

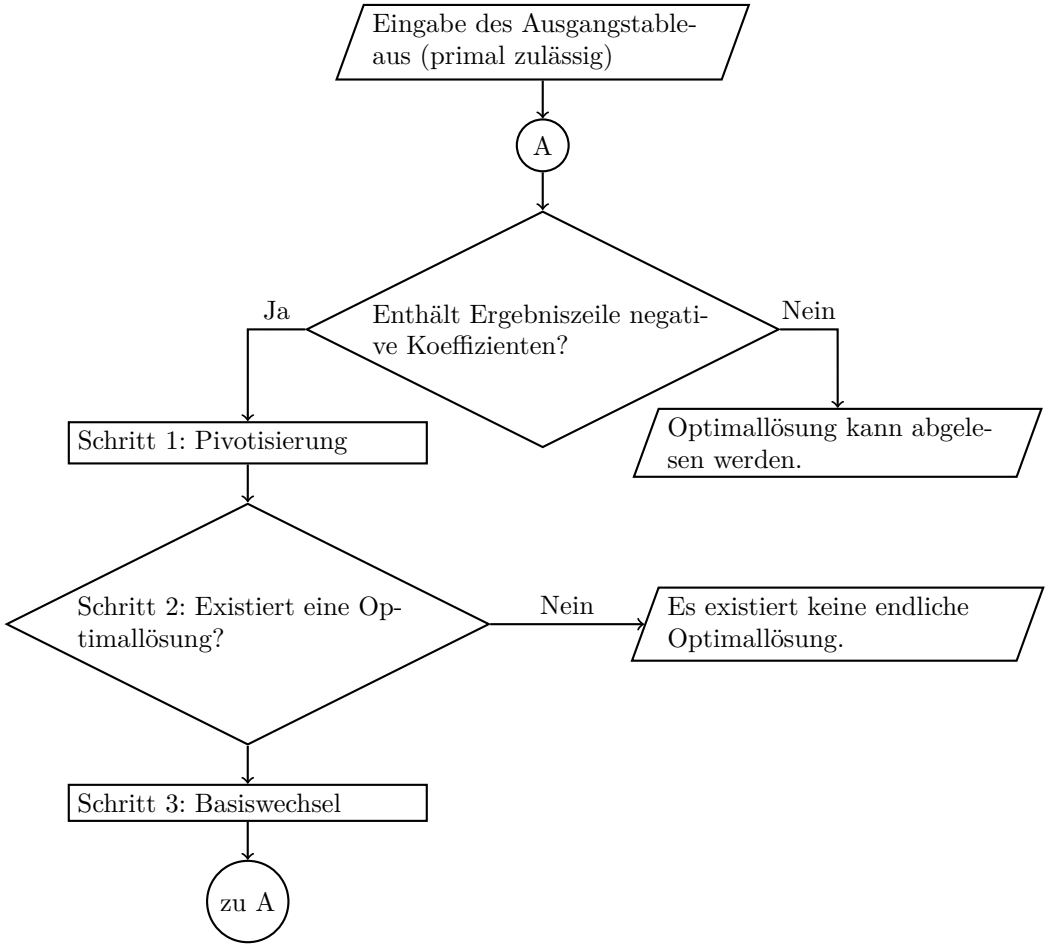

Abb. 10: Vereinfachte Darstellung des Verhaltens des Primal-Simplex-Verfahren
(eigene Darstellung in Anlehnung an Kronfellner/Peschek, 1986, S. 155)

Das Ausgangstableau dieses speziellen Problems liegt nach Einführung von Schlupfvariablen direkt in kanonischer Form vor, welches primal zulässig ist. Ausgehend von der Basislösung $x_1, x_2, ..., x_{n-m} = 0$ wird versucht die Lösung schrittweise zu verbessern, ohne den Zulässigkeitsbereich zu verlassen. Das Verfahren endet, wenn die Zielfunktion optimal ist. (vgl. Koop/Moock, 2008, S. 61 f)
Betrachten wir die 3 Schritte genauer: Pivotisierung, Klärung der Existenzfrage einer Optimallösung und Basiswechsel.

1. Bestimmung eines Pivotelements

   i. Ermittlung jenes j, für das gilt

   $$-c_j^{(r)} = \min\{-c_k^{(r)} : -c_k^{(r)} < 0 \wedge k \in M_{NB}^{(r)}\},$$

   wobei $M_{NB}^{(r)}$ die Indexmenge der Basisvariablen des r-ten Simplextableau ist. j-te Spalte ist die Pivotspalte.

   ii. Ermittlung jenes i, für das gilt

   $$\frac{b_i^{(r)}}{a_{ij}^{(r)}} = \min\{\frac{b_l^r}{a_{lj}^{(r)}} : a_{lj}^{(r)} > 0, l = 1, ..., m\}.$$

   i-te Zeile ist die Pivotzeile.

2. Klärung der Existenzfrage einer Optimallösung

   Ist $a_{lj}^{(r)} \leq 0$ für $l = 1, ..., m$, dann breche das Verfahren ab, denn folglich existiert keine endliche Optimallösung.

3. Durchführung eines Basiswechsels mit dem Pivotelement $a_{ij}^{(r)}$ und Berechnung des $(r+1)$-te Simplextableaus (vgl. Koop/Moock, 2008, S. 63)

   Die genau definierten Schritte zur Erzeugung eines Einheitsvektors möchte ich an dieser Stelle auslassen. Sie sind vergleichbar mit denen des Gauß'schen Algorithmus.

Sonderfälle beim Primal-Simplex-Verfahren

4.6 Dual-Simplex-Verfahren

Da der Primal-Simplex-Algorithmus die Lösung von Linearen Programmen auf spezielle Maximierungsprobleme reduziert, bei dem der Ursprung stets eine zulässige Basislösung ist, wollen wir im Folgenden ein Verfahren zur Lösung von speziellen Minimierungsaufgaben der Form

$$\min z = F(x) = c^T x \tag{3}$$

Sofern in einem optimalen Tableau $c_k = 0$ mit $k \in M_{NB}$ existieren, würde sich der Wert der Zielfunktion nicht ändern, wenn man sie in die Lösung einbezöge. Das bedeutet, dass es in diesem Fall unendlich viele optimale Lösungen gibt, denn die Kante ist parallel zur Zielfunktion. (vgl. Kronfellner/Peschek, 1986, S. 199)

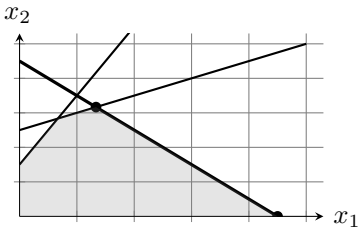

Abb. 11: Simplex: Unendlich viele Optimallösungen
(eigene Darstellung)

Wenn man keinen positiven Wert für $\frac{b_l^{(r)}}{a_{lj}^{(r)}}$ für $l = 1, ..., m$ findet, wobei j die Pivotspalte ist, und damit auch kein Pivotelement, ist die Lösungsmenge unbeschränkt, und es gibt keine endliche Optimallösung. (vgl. Kronfellner/Peschek, 1986, S. 199)

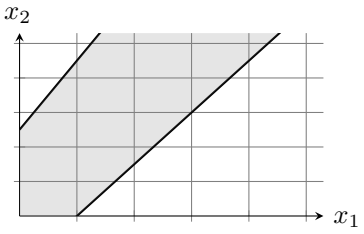

Abb. 12: Simplex: Keine endliche Optimallösung
(eigene Darstellung)

Wenn mehrere minimale Werte $\frac{b_l^{(r)}}{a_{lj}^{(r)}}$ für $l = 1, ..., m$ mit $a_{lj}^{(r)} > 0$ existieren, d.h. das Pivotelement nicht eindeutig bestimmt werden kann, tritt im Folgenden Degeneration auf. Wenigstens ein b_l wird den Wert null annehmen, was in Ausnahmefällen zu Endlosschleifen führen könnte. (vgl. Kronfellner/Peschek, 1986, S. 199)

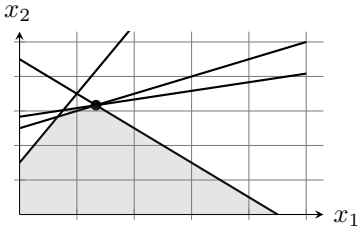

Abb. 13: Simplex: Degeneration
(eigene Darstellung)

$$\textbf{u.d.N.} \sum_{j=1}^{n} a_{ij} x_j \geq b_i$$
$$x_j \geq 0$$
$$\text{mit } c_j \geq 0 \qquad \forall j.$$

einführen.

Das Ausgangstableau hat nach der Transformierung der Zielfunktion in

$$\max z = -F(x) = -c_j x_j \text{ mit } c_j \geq 0 \qquad \forall j.$$

und der Umformung der Ungleichungen in

$$\sum_{j=1}^{n} -a_{ij} \leq -b_i$$

die Gestalt:

BV	x_1	$\cdots$	x_n	x_{n+1}	$\cdots$	x_{n+m}	z	b
x_{n+1}	$-a_{11}$	$\cdots$	$-a_{1n}$	1	$\cdots$	0	0	$-b_1$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
x_{n+m}	$-a_{m1}$	$\cdots$	$-a_{mn}$	0	$\cdots$	1	0	$-b_m$
z	c_1	$\cdots$	c_n	0	$\cdots$	0	1	Z.-Wert

Nun wird versucht unter Wahrung der Dualzulässigkeit $c_j \geq 0$ für $j = 1, ..., n$ durch Pivotieren sich in Richtung Primalzulässigkeit zu bewegen. Wenn nun alle b_i positiv sind, so hat man eine Ecke des Zulässigkeitsbereichs erreicht, die gleichzeitg optimal ist. (vgl. Koop/Moock, 2008, S. 73 ff)

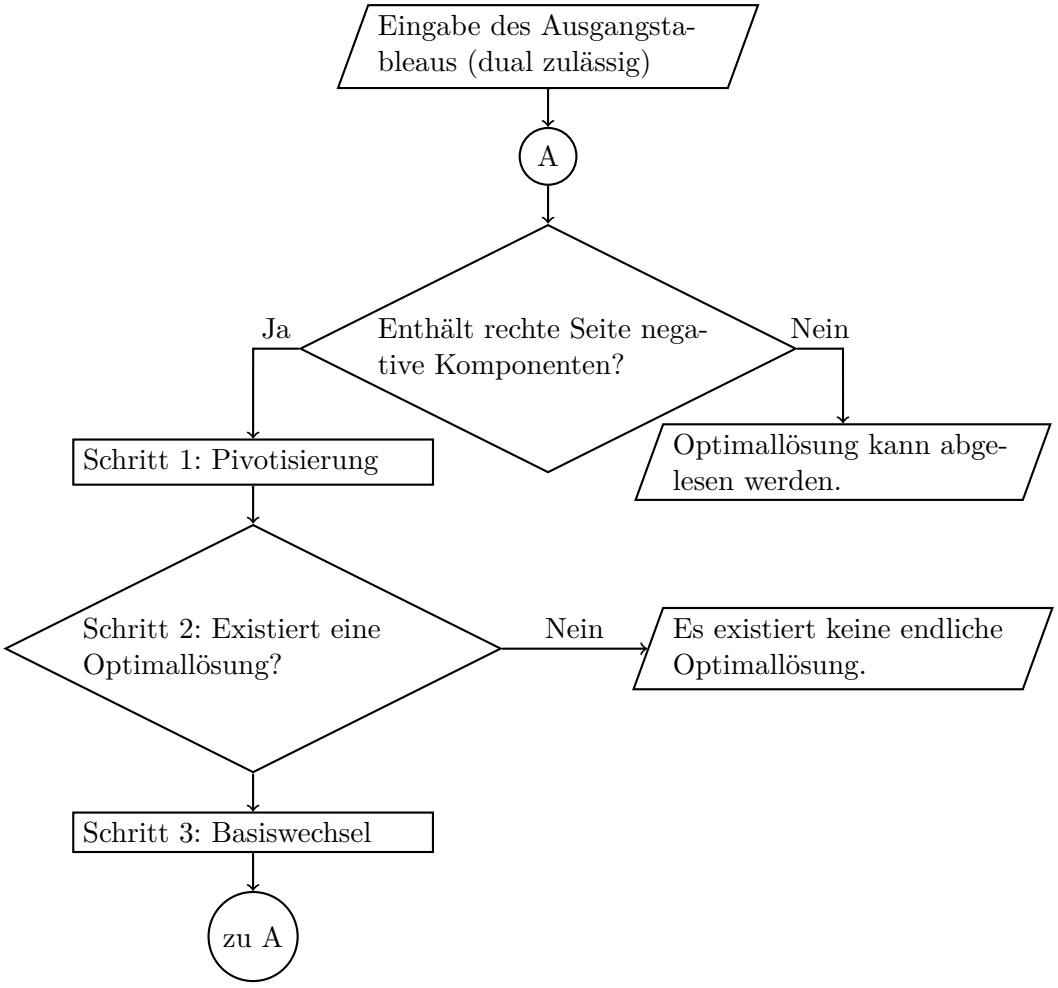

Abb. 14: Vereinfachte Darstellung des Verhaltens des Dual-Simplex-Verfahren
(eigene Darstellung in Anlehnung an Kronfellner/Peschek, 1986, S. 155)

Nun betrachten wir auch die drei Schritte Pivotisierung, Klärung der Existenzfrage einer Optimallösung und Basiswechsel beim Dual-Simplex-Verfahren genauer:

1. Bestimmung eines Pivotelements

   i. Ermittlung jenes i, für das gilt

   $$-b_i^{(r)} = \min\{-b_k^{(r)} : -b_k^{(r)} < 0, k = 1, ..., m\}.$$

   i-te Spalte ist die Pivotzeile.

   ii. Ermittlung jenes j, für das gilt

   $$-\frac{c_j^{(r)}}{a_{ij}^{(r)}} = \min\{-\frac{c_l^{(r)}}{a_{il}^{(r)}} : -a_{il}^{(r)} < 0 \wedge l \in M_{NB}^{(r)}\}.$$

   M_{NB}^r sei eine Indexmenge, die angibt welche Variablen Nichtbasisvariablen sind. j-te Zeile ist die Pivotspalte.

2. Klärung der Existenzfrage einer Optimallösung

   Ist $-a_{il}^{(r)} \geq 0$ für $l = 1, ..., m$, dann breche das Verfahren ab, denn folglich existiert keine endliche Optimallösung.

3. Durchführung eines Basiswechsels mit dem Pivotelement $a_{ij}^{(r)}$ und Berechnung des $(r+1)$-te Simplextableaus (vgl. Koop/Moock, 2008, S. 76)

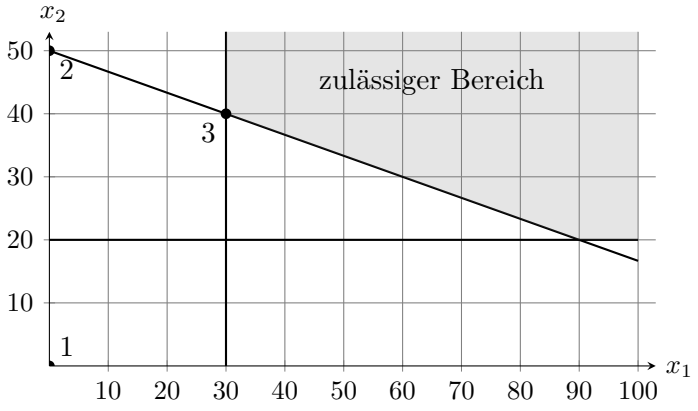

Abb. 15: Lösungsschritte des Beispiels 4
(eigene Darstellung)

4.7 Dualität

Die lineare Zielfunktion wird grundsätzlich von Nebenbedingungen beschränkt, denn ansonsten könnte die Zielfunktion beliebig groß gemacht werden. Diesen Zusammenhang machen wir uns zu Nutze, indem wir eine obere Schranke für die Zielfunktion suchen. Dazu multiplizieren wir die Nebenbedingungen mit einem Fakor λ_i und addieren sie. Wir berechnen also

$$\sum_{i=1}^{m} b_i \lambda_i,$$

wobei dieser Wert durch verschiedene Faktoren verschiedene obere Schranken ergibt. (vgl. Becker, 2014)
Das folgende Beispiel soll uns ein besseres Verständnis geben.

Beispiel 8 (Herleitung einer oberen Schranke)**.** *Gegeben ist das LP des Beispiels 1:*

$$\max z = F(x_1, x_2) = 1000x_1 + 1200x_2$$

$$\begin{aligned} \textbf{u.d.N.}\ x_1 + x_2 &\leq 40 \\ 200x_1 + 600x_2 &\leq 12\ 000 \\ 5x_1 + 10x_2 &\leq 240 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Wir multiplizieren nun die erste Nebenbedingung mit $\lambda_1 = 400$ und die dritte Nebenbedingung mit $\lambda_3 = 80$ und addieren sie:

$$\begin{array}{cccccc} & 400x_1 & + & 400x_2 & \leq & 16\ 000 \\ (+) & 200x_1 & + & 600x_2 & \leq & 12\ 000 \\ (+) & 400x_1 & + & 800x_2 & \leq & 19\ 200 \\ \hline (=) & 1000x_1 & + & 1800x_2 & \leq & 47\ 200 \end{array}$$

Wir haben nun also mit 47 200 eine obere Schranke für die Zielfunktion hergeleitet. Andere Faktoren λ_i können natürlich andere obere Schranken liefern. Ziel ist es nun, die obere Schranke zu minimieren, denn die beste obere Schranke ist die kleinste. Damit die Summe der Nebenbedingungen eine obere Schranke für die Zielfunktion ist, müssen die Koeffizienten von x_j größer oder gleich c_j sein.
Mathematisch formuliert suchen wir also

$$\min z = F(\lambda_1, ..., \lambda_m) = \sum_{i=1}^{m} b_i \lambda_i$$

unter der Nebenbedingung

$$A^T \lambda \geq c \text{ mit } \lambda \geq 0,$$

damit sicher gestellt wird, dass bei der Summe der Nebenbedingungen die Koeffizienten des $x_j \geq c_j$ sind.
In unserem Fall ergibt sich folgendes Problem:

$$\min z = F(\lambda_1, \lambda_2, \lambda_3) = 40\lambda_1 + 12\ 000\lambda_2 + 240\lambda_3$$

$$\begin{aligned} \textbf{u.d.N.}\ \lambda_1 + 200\lambda_2 + 5\lambda_3 &\geq 1000 \\ \lambda_1 + 600\lambda_2 + 10\lambda_3 &\geq 1200 \\ \lambda_1, \lambda_2, \lambda_3 &\geq 0 \end{aligned}$$

Zusammenfassend lässt sich sagen, dass es zu jedem primalen Optimierungsproblem ein korrespondierendes duales Optimierungsproblem gibt, das sich aus den Koeeffizienten der Zielfunktion und den Nebenbedingungen aufstellen lässt.

Definition 5 (Dualität)**.** Gegeben ist das folgende LP (P) als Maximierungsaufgabe (primales lineares Programm). So lautet das zugehörige duale Programm (D) wie folgt (vgl. Becker, 2014):

$$\begin{array}{ll} \max z = F(x) = c^T x & \min z = F(\lambda) = b^T \lambda \\ \textbf{u.d.N.}\ Ax \leq b & \textbf{u.d.N.}\ A^T \lambda \geq c \\ \underbrace{x \geq 0}_{(P)} & \underbrace{\lambda \geq 0}_{(D)} \end{array}$$

Primales System (P)	Duales System (D)
maximiere:	minimiere:
c: Koeffizienten der Zielfunktion	b: rechte Seite der Nebenbedingungen
b: rechte Seite der Nebenbedingungen	c: Koeffizienten der Zielfunktion
Nebenbedingungen:	Variable:
$(Ax)_i \leq b_i$	$\lambda_i \geq 0$
Variable:	Nebenbedingungen:
$x_j \geq 0$	$(A^T)_j \geq c_j$

Tab. 4: Tabelle zur Dualität von Linearen Programmen in Anlehnung an Hackl, o.J., S. 9

Satz 3 (Satz über starke Dualität)**.** *Gegeben sei ein LP in Normalform (P) und das dazugehörige duale Programm (D). Sofern ein Programm unbeschränkt ist, existiert für das andere keine zulässige Lösung. Sofern es aber zulässige Lösungen gibt, stimmen die Optimallösungen überein (vgl. Hackl, o.J., S. 8):*

$$c^T x = b^T \lambda.$$

Beispiel 9 (Praktische Anwendung der Dualität)**.** *Gegeben ist das LP des Beispiels 4:*

$$\min z = F(x_1, x_2) = 450x_1 + 1000x_2$$

$$\begin{aligned} \textbf{u.d.N.}\ x_1 + 3x_2 &\geq 150 \\ x_1 &\geq 30 \\ x_2 &\geq 20 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Aus diesem speziellen Minimierungsproblem ergibt sich ein spezielles Maximierungsproblem durch den Zusammenhang der Dualität, das sich mit dem Primal-Simplex-Verfahren lösen lässt. Wir erhalten folgendes duale Programm:

$$\max z = F(\lambda_1, \lambda_2, \lambda 3) = 150\lambda_1 + 30\lambda_2 + 20\lambda_3$$

$$\begin{aligned} \textbf{u.d.N.}\ \lambda_1 + \lambda_2 &\leq 450 \\ 3\lambda_1 + \lambda_3 &\leq 1000 \\ \lambda_1, \lambda_2, \lambda_3 &\geq 0 \end{aligned}$$

Schließlich erhalten wir mit Hilfe des Primal-Simplex-Verfahren folgendes Endtableau, aus dem wir die Lösung ablesen können:

BV	λ_1	λ_2	λ_3	λ_4	λ_5	z	c
λ_2	0	1	$-\frac{1}{3}$	1	$-\frac{1}{3}$	0	$\frac{350}{3}$
λ_1	1	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	$\frac{1000}{3}$
z	0	0	20	30	40	1	53 500

Die Strukturvariablen des primalen Problems hängen mit den Schlupfvariablen des dualen Problems zusammen und umgekehrt. Das lässt sich auf den Satz vom komplementären Schlupf zurückführen, den wir aber nicht näher betrachten wollen. (vgl. Koop/Moock, 2008, S. 84)
Die Optimallösung lautet also

$$\min 450x_1 + 1000x_2 = 450 \cdot 30 + 1000 \cdot 40 = 53\,500 = \max 150\lambda_1 + 30\lambda_2 + 20\lambda_3 = 150 \cdot \frac{1000}{3} + 30 \cdot \frac{350}{3}.$$

4.8 Überblick der Lösungsmethoden

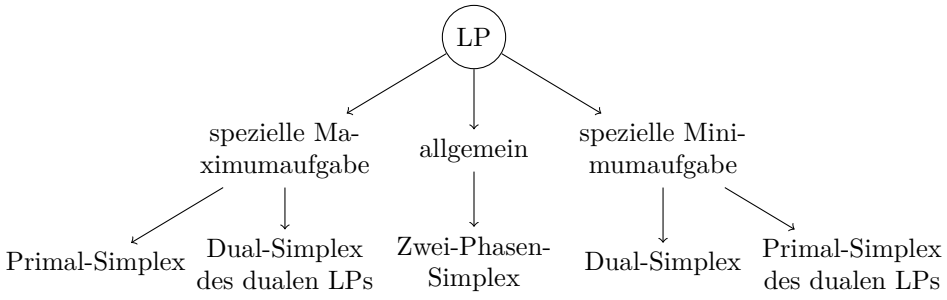

Abb. 16: Überblick der Lösungsmethoden
(eigene Darstellung)

In der Abb. 16 ist ein Überblick der Lösungsmethoden von Linearen Programmen zu sehen. Allgemeine Lineare Programme lassen sich mit dem Zwei-Phasen-Algorithmus lösen. Zuerst wird durch Lösung eines Hilfsproblems nach einer zulässigen Basislösung gesucht und anschließend wird das Problem mit dem Primal-Simplex-Verfahren gelöst, sofern eine endliche Optimallösung existert. (vgl. Leydold, 1997)

5

Resümee

Zusammenfassend ist zu sagen, dass die Kenntnis von einfachen Mitteln der Mathematik, die laut mehreren Schüler*innen für ihren Alltag scheinbar sinnlos erscheinen, brauchbar und zweckmäßig ist, wie das Tool der Linearen Optimierung zeigt. Aus meiner Sicht ist es erstrebenswert, diesen Bereich wieder in den Mathematiklehrplan der AHS zu implementieren, denn zahlreiche Argumente sprechen dafür:

- Bei Linearen Programmen, die auf zwei Variablen beschränkt sind, ist es bereits mit geringen Aufwand möglich, Schüler*innen die Methoden der Linearen Optimierung zu vermitteln.
- Schüler*innen erfahren, wie konkret Mathematik in der Wirtschaft eingesetzt wird.
- Schüler*innen kann gezeigt werden, dass eine fundierte Kenntnis von *linearen Ungleichungen und Gleichungen* sinnvoll ist.
- Die Anwendung der Linearen Optimierung bietet die Möglichkeit, Schüler*innen begreifbar zu machen, dass Mathematik sehr vielfältigen Einsatz findet und die Basis für viele Abläufe bildet.
- Praxisnahe Anwendungen erleichtern das Lernen und können das Interesse an der Mathematik fördern.

Abbildungsverzeichnis

Literaturverzeichnis

Becker, P. (2014): Dualität. URL: http://www2.inf.h-brs.de/~pbecke2m/or1/dual1.pdf [Zugriff: 6.1.2019].

Blaha, M. (o.J.): Ungleichungen, Ungleichungssysteme. URL: https://www.mathe-online.at/nml/materialien/SkriptumBlaha/KAP-03.pdf [Zugriff: 24.8.2018].

Brünner, A. (o.J.): Zitate über Mathematik. URL: http://www.arndt-bruenner.de/mathe/Allgemein/zitate.htm [Zugriff: 2.11.2018].

Burger, M. (2006): Mathematische Modellierung. Vorlesungskriptum. Münster. URL: https://www.uni-muenster.de/AMM/num/Vorlesungen/Modellierung_06/skript.pdf [Zugriff: 22.7.2018].

Burkard, R. (o.J.): Einführung in die Mathematische Optimierung. Vorlesungsskriptum. Graz. URL: https://www.opt.math.tugraz.at/~hatzl/Vorlesungen/MathoptSS09/Opt.pdf [Zugegriff: 25.8.2018].

Dudenredaktion (o.J.): "Polyeder" auf Duden online. URL: https://www.duden.de/node/717441/revisions/1136408/view [Zugriff: am 26.8.2018].

Embacher, F./Oberhuemer, P. (o.J.): Matrizen. URL: https://www.mathe-online.at/mathint/matr/i.html [Zugriff: 03.9.2018].

Hackl, B. (o.J.): Lineare Optimierung. Theoriezusammenfassung kompakt. Version 1.00 [Anhang einer Email von Phillip Hungerländer, 11.7.2018].

Kaufmann, J. (1991): *Algebra with Trigonometry for College Students*. Boston: PWS Publishers.

Keil, D. (o.J.): Cramer'sche Regel. URL: http://www.abi-mathe.de/buch/matrizen/cramersche-regel/ [Zugriff: 13.8.2018].

Koop, A./Moock, H. (2008): *Lineare Optimierung: Eine anwendungsorientierte Einführung in Operations Research*. Berlin Heidelberg: Spektrum Akademischer Verlag.

Kronfellner, M./Peschek, W. (1985): *Angewandte Mathematik 1*. Wien: Verlag Hölder-Pichler-Tempsky.

Kronfellner, M./Peschek, W. (1986): *Angewandte Mathematik 3*. Wien: Verlag Hölder-Pichler-Tempsky.

Leydold, J. (1997): 2-Phasen-Simplex-Algorithmus. URL: http://statistik.wu-wien.ac.at/~leydold/MOK/HTML/node164.html [Zugriff: 6.1.2019].

Makhorin, A. (2014): GLPK (GNU Linear Programming Kit). URL: https://www.gnu.org/software/glpk/ [Zugriff: 5.1.2019].

Scheid, H./Schwarz, W. (2009): *Elemente der Linearen Algebra und der Analysis*. Berlin Heidelberg: Spektrum Akademischer Verlag.

Schneider, A. (o.J.a): Lineare Optimierung. URL: https://www.mathebibel.de/lineare-optimierung [Zugriff: 24.8.2018].

Schneider, A. (o.J.b): Simplex-Algorithmus. URL: https://www.mathebibel.de/simplex-algorithmus [Zugriff: 29.12.2018].

Schwarz, A. (o.J.): Lineare Optimierung. Lehrbuch mit Aufgaben und Lösungen. URL: https://www.mathe-aufgaben.com/uploads/media/Leseprobe_Lineare_Optimierung.pdf [Zugriff: 5.11.2018].

Seiffart, E./Manteuffel, K. (1974): *Lineare Optimierung*. Leipzig: BSB B.G. Teubner Verlagsgesellschaft.

Sonne, B./Weiß, R. (2013): *Einsteins Theorien: Spezielle und Allgemeine Relativitätstheorie für interessierte Einsteiger und zur Wiederholung*. Berlin Heidelberg: Spektrum Akademischer Verlag.

Stobitzer, C. (o.J.): Obere und Untere Schranke / Supremum und Infimum. URL: http://www.mathepauken.de/schranken-obere-untere-supremum-infimum.php [Zugriff: 4.11.2018].

TutorCircle-Editorship (o.J.): Cramer's Rule. URL: http://math.tutorcircle.com/algebra-2/cramers-rule.html#proof [Zugriff: 22.7.2018].

Wohlgemuth, M. (2009): *Mathematisch für Anfänger*. Berlin Heidelberg: Spektrum Akademischer Verlag.

A

Mathematische Grundlagen

Im Anhang A möchte ich noch mathematische Grundlagen ansprechen, die für komplexere Aufgaben der Linearen Optimierung von besonderer Wichtigkeit sind. Insbesondere wird auf die Algorithmen zur Lösung von linearen Gleichungssysteme eingegangen, die die Basis für Lösungsverfahren von Optimierungsproblemen bilden.

A.1 Matrizenschreibweise

Ein zentrales Element zur Lösung und Darstellung von linearen Zusammenhängen sind Matrizen, deren Vorteil es ist, LGS übersichtlich darzustellen. Weiters sind Konstrukte wie Determinante und Inverse einer Matrix Instrumente zur Angabe von Lösungsfällen. (vgl. Embacher/Oberhuemer, o.J.)

Definition 6. Unter einer Matrix versteht man ein rechteckiges Zahlenschema, das allgemein aus n Zeilen und m Spalten besteht, welche damit definitionsgemäß eine $m \times n$-Matrix (sprich: m mal n-Matrix) ist. Die $n \cdot m$ vorkommenden Elemente einer Matrix nennt man Komponenten der Matrix:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}. \quad \text{(vgl. Embacher/Oberhuemer, o.J.)}$$

Die transponierte Matrix entsteht durch Vertauschung der Zeilen- und Spalten der eigentlichen Matrix $A = (a_{ij}) \in \mathbb{R}^{n \times m}$. Diese neue Matrix wird als $A^T = (a'_{ji}) \in \mathbb{R}^{m \times n}$ notiert, wobei für $a'_{ji} = a_{ij}$ für $j = 1, 2, \ldots, n$ und für $i = 1, 2, \ldots, m$ gilt. Ein Vektor $x \in \mathbb{R}^n$ kann also als eine $n \times 1$-Matrix definiert werden. Somit kann das Skalarprodukt zweier Vektoren $x, y \in \mathbb{R}^n$ auch in der Form $x^T y$ angeschrieben werden. (vgl. Koop/Moock, 2008, S. 18)

A.2 Lineare Gleichungssysteme

Definition 7. Nach der Definition von Harld Scheid und Wolfgang Schwarz nennt man eine algebraische Gleichung linear, wenn die Variablen $x_1, x_2, ..., x_n$ nur in der ersten Potenz und nicht als Produkte vorkommen. Somit ist die Gleichung der Art $a_1x_1 + a_2x_2 + ... + a_nx_n = b$ bzw. $\sum_{j=1}^{n} a_jx_j = b$. (vgl. Scheid/Schwarz, 2009, S. 3)
Sind mehrere algebraische Gleichungen ersten Grades gegeben, deren mathematische Korrektheit gleichzeitig erfüllt sein sollte, spricht man von einem linearen Gleichungssystem. Ein lineares Gleichungssystem mit n Variablen und m Gleichungen besitzt die Form:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n &= b_m. \end{aligned}$$

(vgl. Scheid/Schwarz, 2009, S. 3)

Die Koeffizienten des Systems sind $a_{ij}, b_i \in \mathbb{R}$ und $x_j \in \mathbb{R}$ die Unbekannten. Das obige System kann nun mithilfe des Summenzeichens verkürzt dargestellt werden:

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

Wenn wir zusätzlich die Systemmatrix (Koeffizientenmatrix) $A \in \mathbb{R}^{m\times n}$ und die Vektoren $x \in \mathbb{R}^n, b \in \mathbb{R}^m$ durch

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

definieren, erlaubt diese Matrixschreibweise, das System in die vereinfachte Form **Ax=b** zu bringen. (vgl. Koop/Moock, 2008, S. 18 f)
Zusätzlich wird ein solches Gleichungssystem als homogen bezeichnet, wenn für alle $i \in \{1, 2..., m\}$ gilt: $b_i = 0$. Somit wäre null (für alle Unbekannten x_j) eine Lösung des Systems. Andernfalls heißt es inhomogen. (vgl. Wohlgemuth, 2009, S. 95)

A.3 Transformation eines linearen Ungleichungssystems in ein LGS

Bei der Lösung linearer Ungleichungssysteme in mehr als zwei Variablen wird das System in ein LGS umgewandelt. Dazu werden sogenannte Schlupfvariablen eingeführt, was bei Linearen

Programmen dazu führt, dass außer den Nichtnegativitätsbedingungen nur Gleichungen auftreten. (vgl. Scheid/Schwarz, 2009, S. 156)
Nebenbedingungen des Beispiels 1 als transformiertes LGS:

$$\begin{aligned} x_1 + x_2 \quad & +x_3 \quad & = 40 \\ 200x_1 + 600x_2 \quad & + x_4 \quad & = 12\ 000 \ \ (x_1, x_2, x_3, x_4, x_5, x_6 \geq 0) \\ 5x_1 + 10x_2 \quad & & +x_5 = 240. \text{ (vgl. Scheid/Schwarz, 2009, S. 156)} \end{aligned}$$

A.4 Lösbarkeit linearer Gleichungssysteme

Nach der Definition linearer Gleichungssysteme und der Notation, stellt sich jetzt die Frage, wie man die Lösungsmenge eines solchen Systems bestimmt. Bereits in der Schule hat man verschiedene Verfahren gelernt, wie das Additionsverfahren, das Einsetzverfahren oder das Gleichsetzungsverfahren – um einige zu nennen. Für Gleichungssysteme größerer Dimension werden solche Verfahren aber schnell zu langwierig. Die Idee ist es nun, die gegebenen Verfahren zu formalisieren bzw. zu generalisieren, sodass jeder Schritt allgemein festgelegt ist und falls notwendig oder erwünscht auch programmiert werden kann. (vgl. Wohlgemuth, 2009, S. 97)

A.4.1 Gauß'sche Algorithmus

> Der Gauß'schen Algorithmus macht sich das Prinzip zu Nütze, dass elementare Umformungen zwar das System verändern, aber die Lösung erhalten bleibt. Ziel dieser Methode ist es, das Schema so umzuformen, dass die Lösung möglichst einfach eruierbar ist. Alle komplexeren linearen Optimierungsprobleme basieren auf dieser Anwendung. (vgl. Koop/Moock, 2008, S. 24)

Es liegt ein beliebiges lineares Gleichungsystem vor. (vgl. Koop/Moock, 2008, S. 24)

x_1	x_2	$\cdots$	x_n	
a_{11}	a_{12}	$\cdots$	a_{1n}	b_1
a_{21}	a_{22}	$\cdots$	a_{2n}	b_2
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
a_{m1}	a_{m2}	$\cdots$	a_{mn}	b_m

Die folgenden Operationen ändern die Lösungsmenge des Systems nicht und sind somit zulässig:

- Vertauschung von Zeilen oder Spalten
- Multiplikation einer Zeile mit einem von null verschiedenen Element $\lambda \in \mathbb{R} \setminus \{0\}$
- Addition des Vielfachen einer Zeile zu einer anderen Zeile (vgl. Koop/Moock, 2008, S. 24)

Vorgehensweise:

1. Wir bezeichnen die Spalten der Matrix mit $A_1, A_2, A_3, ..., A_n$. Wir prüfen, ob der oberste Eintrag der ersten Spalte ungleich null ist. Tritt dies nicht ein, vertauschen wir Zeilen, sodass der oberste Eintrag ungleich null ist, um eine Division durch null zu vermeiden.
2. Nun nehmen wir an, dass $a_{11} \neq 0$ ist und bringen alle Einträge unter a_{11} auf null, indem wir iterativ das $(-1) \cdot \frac{a_{i1}}{a_{11}}$-fache der ersten Zeile zur i-ten Zeile addieren. Die erweiterte Koeffizientenmatrix besitzt jetzt eine solche Gestalt:

$$\left(\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{m2}^{(2)} & \cdots & a_{mn}^{(2)} & b_m^{(2)}. \end{array}\right)$$

3. All jene Einträge $a_{ij}^{(2)}, b_i^{(2)}$ mit $i = \{2, 3, ..., m\}$ und $j = \{2, 3, ..., n\}$ sind diejenigen Komponenten, die sich nach Schritt 1 verändert haben. Nun betrachten wir vorübergehend die Teilmatrix $A^{(2)}|b^{(2)}$ mit den Spalten $A_1^{(2)}, A_2^{(2)}, ..., A_{n-1}^{(2)}$ und wiederholen analog bei Schritt 2.
4. Dieses Verfahren setzten wir fort, bis entweder in einer entstehenden Teilmatrix nur mehr Nullspalten existieren oder wir die letzte Zeile bzw. Spalte erreicht haben. (vgl. Wohlgemuth, 2009, S. 98 ff)

Ergebnis dieser Zeilenumformung sollte sein, dass jede Zeile der Koeffizientenmatrix, welche keine Nullzeile ist, mit je einer null mehr beginnt. Eine Systemmatrix dieser Art bezeichnet man als Zeilenstufenform. Jener Eintrag der Zeilen, der von null verschieden ist, stellt das Pivotelement dar. (vgl. Wohlgemuth, 2009, S. 100)

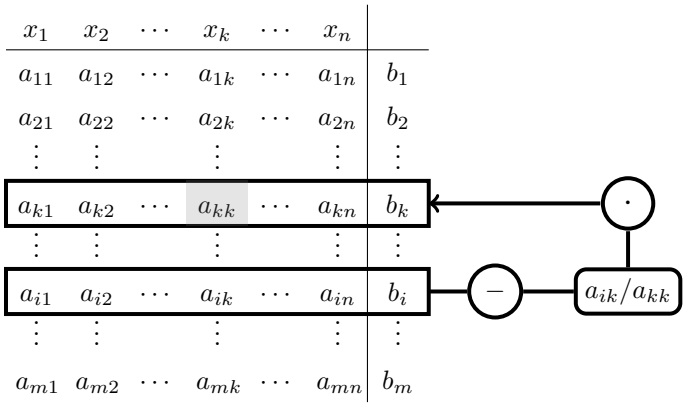

Abb. 17: Illustration zum Gauß'schen Algorithmus
(eigene Darstellung in Anlehnung an Koop/Moock, 2008, S. 27)

A.4.2 Lösungsfälle linearer Gleichungssysteme

Im nächsten Abschnitt versuchen wir die Lösungsfälle genau zu definieren und zu beschreiben. Dabei wissen wir schon von der Schule, dass ein lineares Gleichungssystem entweder eine, keine

oder unendlich viele Lösungen haben kann. Um das Eintreten der Lösungsfälle genau bestimmen zu können, müssen wir zuvor noch zwei Begriffe einführen: die lineare Abhängig- bzw. Unabhängigkeit und den Rang einer Matrix.

Definition 8. Laut Harald Scheid und Wolfang Schwarz heißt eine Menge A von Vektoren aus einem Vektorraum V linear unabhängig, wenn eine Gleichung der Form $r_1\vec{a_1}+r_2\vec{a_2}+...+r_n\vec{a_n} = \vec{0}$ mit $\{a_1, a_2, ..., a_n\} \subseteq A$ nur mit $r_1 = r_2 = ... = r_n = 0$ zu einer mathematisch korrekten Aussage führt. Anderfalls nennt man die Vektormenge linear abhängig. (vgl. Scheid/Schwarz, 2009, S. 14) Daraus folgt, dass sich bei linearer Abhängigkeit mindestens ein Vektor aus Linearkombinationen der anderen berechnen lässt. (vgl. Koop/Moock, 2008, S. 20)

Definition 9. Der Rang $r(A)$ einer Matrix $A \in \mathbb{R}^{m\times n}$ zählt die maximale Anzahl der linear unabhängigen Spaltenvektoren von A. (vgl. Koop/Moock, 2008, S. 22)

Dabei äquivaliert die Anzahl der linear unabhängigen Zeilenvektoren bzw. der linear unabhängigen Spaltenvektoren der transponierten Matrix A^T mit der Anzahl der linear unabhänigen Spaltenvektoren von A. (vgl. Koop/Moock, 2008, S. 22) Um den Rang der Koeffizientenmatrix zu bestimmen, zählt man die Anzahl der nullverschiedenen Zeilen der Matrix in Zeilenstufenform. (vgl. Wohlgemuth, 2009, S. 110)

Diese Begriffe helfen uns nun, die Lösungsfälle eines Gleichungssystems ersten Grades genau zu charakterisieren. Dazu definieren wir vorerst die erweiterte Koeffizientenmatrix:

$$A|b = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_n \end{pmatrix} \in \mathbb{R}^{m\times(n+1)}.$$

Satz 4. *„Sei $A \in \mathbb{R}^{m\times n}$,$b \in \mathbb{R}^m$. Dann ist das Gleichungssystem $Ax = b$ genau dann lösbar, wenn $r(A) = r(A|b)$. Gilt zusätzlich $r(A) = n$, so gibt es genau eine Lösung $x \in \mathbb{R}^n$. Im Falle $r(A) < n$ gibt es unendlich viele Lösungen mit $n - r(A)$ freien Parametern.“ (Koop/Moock, 2008, S. 22)*

Gibt es nun mehr Variablen als Gleichungen ($m < n$), kann man schon sagen, dass das System keine eindeutige Lösung besitzt. (vgl. Koop/Moock, 2008, S. 23)

A.4.3 Determinante zur Lösung von LGS und Bestimmung des Ranges

Die Determinante und die Cramer'sche Regel, die im Folgenden beschrieben werden, sollen die Zusammenhänge der Algebra aufzeigen und Ausblick über die Möglichkeiten zur Lösung von LGS geben.

Definition 10. Die Determinante ist eine Zahl (Skalar), die einer quadratischen Matrix ($m = n$) zugeordnet wird und aus ihren Komponenten berechnet wird. Die Determinante einer 3×3-Matrix A ist definiert durch

$$det\,(A) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

$$= a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{31}a_{22}a_{13} -$$

$$a_{32}a_{23}a_{11} - a_{33}a_{21}a_{12}.\text{ (vgl. Kaufmann, 1991, S. 582 ff)}$$

Satz 5 (Cramer'sche Regel)**.** *Sei $Ax = b$ ein lineares Gleichungssystem, $A \in \mathbb{R}^{m \times n}$ mit $m = n$ eine reguläre Matrix (invertierbar; sprich $det\,(A) \neq 0$) und seien $b \in \mathbb{R}^m$ die konstanten Glieder des Systems, dann hat das System die eindeutige Lösung*

$$x_i = \frac{1}{det\,(A)} \cdot det\,(A_j)\,.\text{ (vgl. Keil, o.J.)}$$

Somit kann man mithilfe der Cramer'schen Regel bei $det\,(A) = 0$ angeben, dass das LGS keine eindeutige Lösung besitzt und mit anderen Methoden bestimmen, ob es unter- oder überstimmt ist. Analog kann man bei $det\,(A) \neq 0$ die Lösung explizit angeben. (vgl. Kaufmann, 1991, S. 585)

Lemma 1. Laut Wohlgemuths Definition gilt für die Multiplikation zweier Determinanten: $det\,(AB) = det\,(A) \cdot det\,(B)$. (vgl. Wohlgemuth, 2009, S. 141)

Beweis 2. Gegeben ist ein lineares Gleichungssystem $Ax = b$ mit $A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ und $m = n$. Betrachtet wird ein LGS, dessen $det\,(A) \neq 0$ ist, was impliziert, dass das System eine eindeutige Lösung besitzt.

Man ersetzt die j-te Spalte der Einheitsmatrix mit dem Lösungsvektor x und nennt diese Matrix X_j. Damit ist $AX_j = A_j$, wobei die j-te Spalte der Koeffizientenmatrix durch b ersetzt wurde.

Da $det\,(X_j) = x_j$ ist, folgt

$$det\,(A) \cdot det\,(X_j) = det\,(A_j)$$
$$x_j = det\,(A)^{-1} \cdot det\,(A_j)\,.$$

Da die Matrix A nach Voraussetzung regulär ist, existiert $det\,(A)$. (vgl. TutorCircle-Editorship, o.J.) □

B

Programm

Im Anhang B wird der in der Arbeit angesprochenen Algorithmus zur Lösung von LGS als C++-Programm dargestellt. Auf die Optimierung hinsichtlich Effizienz und Geschwindigkeit wurde verzichtet. Es handelt sich vielmehr um einen Versuch, besprochene Verfahren mit dem Computer umzusetzen und zu generalisieren, um komplexere Aufgaben zu lösen.

B.1 Gauß'scher Algorithmus

```
1  //
   //-PARAMETER-------------------------------------------------------------------
3  // Typ                              Name          Beschreibung
   //-----------------------------------------------------------------------------
5  // int                              m             Anzahl der Gleichungen
   // int                              n             Anzahl der Variablen
7  // int                              k             Pivotindex
   // min                              min           Minimum unter m und n
9  // int                              r             Rang der Matrix
   // int                              maxindex      Index des Maximums
11 // double                           max           Maximum
   // vector <double>                  swap          Hilfsvektor
13 // vector <vector <double> >        matrix        erweiterte Koeffizientenmatrix

15
   #include <iostream>
17 #include <vector>

19 using namespace std;

21 void printmatrix(vector <vector <float> > pmatrix){
       for(int i=0;i<pmatrix.size();i++){
23         for(int j=0;j<pmatrix[i].size();j++){
               cout<<pmatrix[i][j]<<" ";
25             if(j==(pmatrix[i].size()-2)){
                   cout<<"| ";
```

```
27              }
            }
29          cout<<endl;
        }
31  }

33  void rangmatrix(vector <vector <float> > rmatrix){
        int r=0,h=0;
35      for(int i=0;i<rmatrix.size();i++){
            for(int j=0;j<(rmatrix[i].size()-1);j++){
37              if(rmatrix[i][j]==0){
                    h++;
39              }
            }
41          if(h!=(rmatrix[i].size()-1)){
                r++;
43          }
            h=0;
45      }

47      cout<<"r(A) = "<<r<<endl;
        r=0;
49
        for(int i=0;i<rmatrix.size();i++){
51          for(int j=0;j<rmatrix[i].size();j++){
                if(rmatrix[i][j]==0){
53                  h++;
                }
55          }
            if(h!=rmatrix[i].size()){
57              r++;
            }
59          h=0;
        }
61
        cout<<"r(A|b) = "<<r<<endl;
63  }

65  int main(int argc, const char * argv[]) {

67      vector<vector <float> >matrix;
        int n,m;
69      float component;

71      // Eingabe der erweiterten Koeffizientenmatrix
        cout<<"Geben Sie die Anzahl der Variablen und Gleichungen Ihres LGS ein:"<<
        endl<<"Anzahl der Variablen; n = ";
73      cin>>n;

75      cout<<"Anzahl der Gleichungen; m = ";
```

```
    cin>>m;
77
    cout<<endl<<"Geben Sie die erweiterte Koeffizientenmatrix jeweils zeilenweise
    ein:"<<endl;
79  for(int i=0;i<m;i++){
        vector<float>temp;
81      for(int j=0;j<=n;j++){
            if(j==n){
83              cout<<"Komponent b_"<<(i+1)<<" = ";
            }
85          else{
                cout<<"Komponent a_"<<(i+1)<<(j+1)<<" = ";
87          }
            cin>>component;
89          temp.push_back(component);
        }
91      cout<<endl;
        matrix.push_back(temp);
93  }

95  cout<<endl<<"Ausgangsmatrix:"<<endl;
    printmatrix(matrix);
97
    // Gauss
99  int k=0, i=k, maxindex,min;
    float max;
101 vector <float> swap(n);

103 if(m<n){
        min=m;
105 }
    else{
107     min=n;
    }
109
    while(k<min){
111
        // Pivotsuche (Maxiumstrategie)
113     max=matrix[k][k];
        maxindex=k;
115     i=k+1;
        while(i<m){
117         if(max<matrix[i][k]){
                max=matrix[i][k];
119             maxindex=i;
            }
121         i++;
        }
123
        if(max!=0){
```

```
125
                // Zeilenvertauschung
127             swap=matrix[maxindex];
                matrix[maxindex]=matrix[k];
129             matrix[k]=swap;

131             // Berechnungen
                i=k+1;
133             while(i<m){
                    for(int j=0;j<=n;j++){
135                     swap[j]=(-1)*(matrix[i][k]/matrix[k][k])*matrix[k][j];
                    }
137                 for(int j=0;j<=n;j++){
                        matrix[i][j]=matrix[i][j]+swap[j];
139                 }
                    i++;
141             }}
            k++;
143     }

145     cout<<endl<<"Trapezgestalt:"<<endl;
        printmatrix(matrix);
147
        cout<<endl<<endl;
149
        rangmatrix(matrix);
151
        return 0;
153 }
```

Listing 1: Gauß'scher Algorithmus

Dieses angegebene Programm transformiert nach Eingabe des LGS die Systemmatrix in Zeilenstufenform und berechnet den Rang der Matrix. In der Abbildung 18 ist das Verfahren anhand eines Flowcharts bildlich dargestellt.

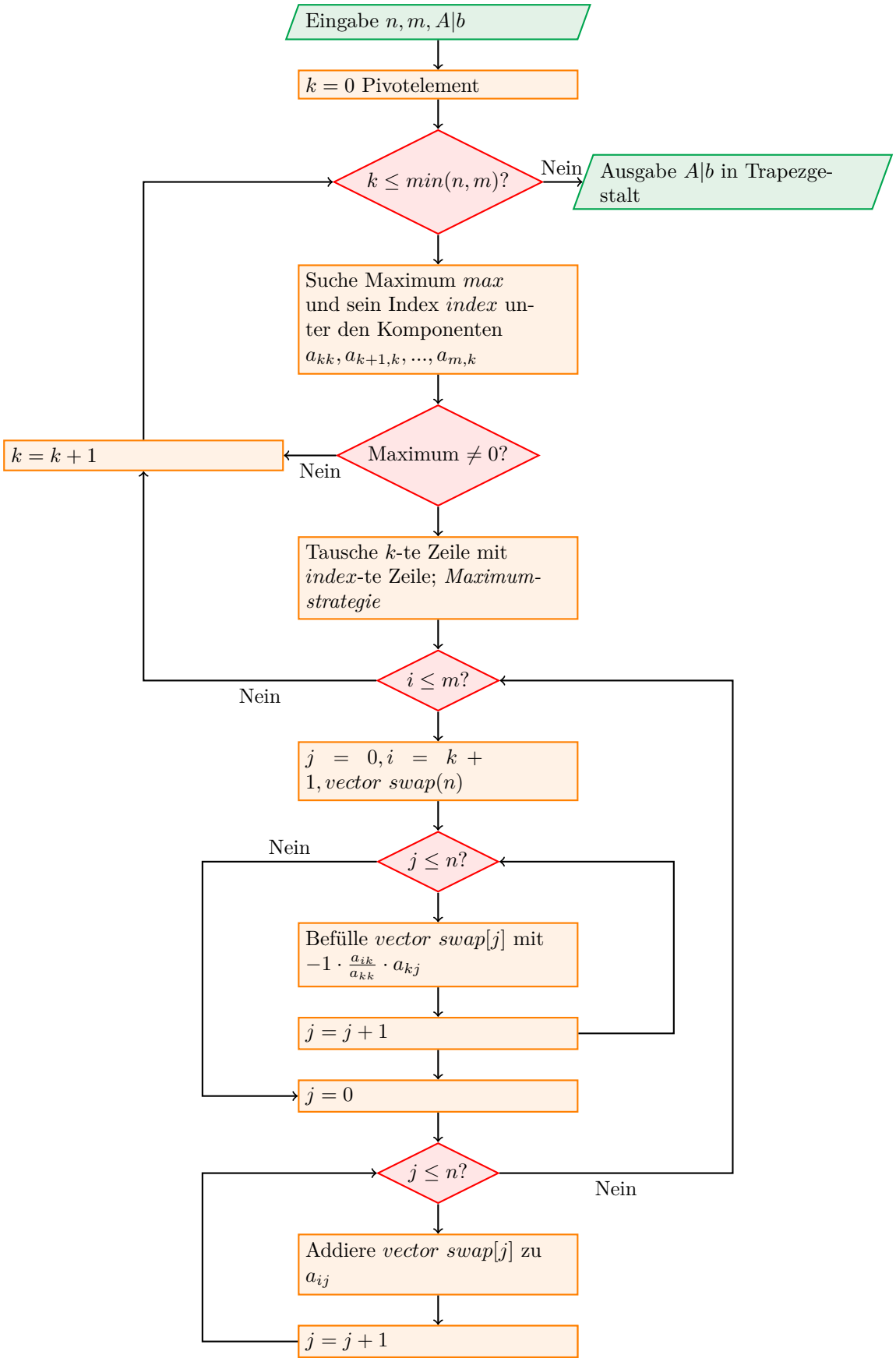

Abb. 18: Flowchart des Gauß'schen Algorithmus
(eigene Darstellung)

B.2 Primal-Simplex-Verfahren

Natürlich handelt es sich hierbei nicht um einen wirklich in der Praxis einsetzbaren Solver, trotzdem ist es cool, sein eigens programmiertes Programm zu haben. Hierfür würde ich gerne auf das GNU Linear Programming Kit verweisen (kurz: GLPK), das eine freie Nutzung erlaubt und es ermöglicht, Probleme aus der Linearen Optimierung und der Ganzzahligen Linearen Optimierung zu lösen. Diese dynamische Programmbibliothek ist auch für größere Probleme aus der Praxis denkbar. (vgl. Makhorin, 2014)

```
1  //
   //-PARAMETER-------------------------------------------------------------------
3  // Typ                              Name          Beschreibung
   //-----------------------------------------------------------------------------
5  // int                              m             Anzahl der Ungleichungen
   // int                              n             Anzahl der Variablen
7  // int                              s             Index der Pivotspalte
   // int                              z             Index der Pivotzeile
9  // int                              r             r-tes Tableau
   // float                            pivot         Pivotelement
11 // vector <float>                   basis         Index der Basisvariablen
   // vector <float>                   ziel          Zielfunkion
13 // vector <vector <float> >         matrix        erweiterte Koeffizientenmatrix

15
   #include <iostream>
17 #include <vector>

19 using namespace std;

21 void printtableau(vector <vector <float> > pmatrix, vector <float> pbasis, vector
       <float> pziel){
    cout<<"BV |";
23  for(int i=0;i<(pziel.size()-1);i++){
        cout<<" x_"<<(i+1);
25  }
    cout<<" | b"<<endl;
27
    for(int i=0;i<pmatrix.size();i++){
29      cout<<"x_"<<(pbasis[i]+1)<<" | ";
        for(int j=0;j<pmatrix[i].size();j++){
31          cout<<pmatrix[i][j]<<" ";
            if(j==(pmatrix[i].size()-2)){
33              cout<<"| ";
            }
35      }
        cout<<endl;
37  }

39  cout<<"z |";
```

```
    for(int i=0;i<pziel.size();i++){
41      if(i==(pziel.size()-1)){
            cout<<" | "<<pziel[i];
43      }
        else{
45          cout<<" "<<pziel[i];
        }
47  }
}
49
int main(int argc, const char * argv[]) {
51  vector<vector <float> >matrix;
    int n,m,e=0,s,z,r=0,h=0;
53  float component,factor,pivot;

55  cout<<"---PRIMAL-SIMPLEX-ALGORITHMUS---"<<endl<<endl;
    cout<<"Zur Loesung von speziellen Maximumproblemen"<<endl<<endl;
57  cout<<"Geben Sie die Anzahl der Variablen und Nebenbedingungen Ihres Linearen
    Programms ein:"<<endl<<"Anzahl der Variablen; n = ";
    cin>>n;
59  cout<<"Anzahl der Nebenbedingungen; m = ";
    cin>>m;
61
    vector <float> ziel (n+m+1);
63  cout<<endl<<"Geben Sie die Koeffizienten der Zielfunktion in der Form max
    c_1x_1+...+c_nx_n ein:"<<endl;
    for(int i=0;i<n;i++){
65      cout<<"Komponent c_"<<(i+1)<<" = ";
        cin>>component;
67      if(component!=0){
            ziel[i]=component*(-1);
69      }
        else{
71          ziel[i]=component;
        }
73  }
    for(int i=n;i<=(n+m);i++){
75      ziel[i]=0;
    }
77



79


81  cout<<endl<<"Geben Sie die Koeffizienten der Nebenbedingungen in der Form u.d.
    N: (Ax)_i<=b:"<<endl;
    for(int i=0;i<m;i++){
83      vector<float>temp;
        for(int j=0;j<=(n+m);j++){
85          if(j==(n+m)){
                cout<<"Komponent b_"<<(i+1)<<" = ";
```

```
87                  cin>>component;
                    temp.push_back(component);
89              }
                else if(j>=n){
91                  if(j==(n+i)){
                        component=1;
93                      temp.push_back(component);
                    }
95                  else{
                        component=0;
97                      temp.push_back(component);
                    }
99              }
                else{
101                 cout<<"Komponent a_"<<(i+1)<<(j+1)<<" = ";
                    cin>>component;
103                 temp.push_back(component);
                }
105         }
            cout<<endl;
107         matrix.push_back(temp);
        }
109
        vector<float> basis (m);
111     for(int i=0;i<m;i++){
            basis[i]=n+i;
113     }
        cout<<endl<<"Ausgangstableau:"<<endl;
115     printtableau(matrix,basis,ziel);

117     // Koeffizienten der Zielfunktion negativ?
        component=ziel[0];
119     for(int j=0;j<(n+m);j++){
            if(ziel[j]<0){
121             e=e+1;
            }
123     }

125     while(e!=0){
            e=0;
127         // Pivotisierung der Spalte
            component=ziel[0];
129         s=0;
            for(int j=0;j<(n+m);j++){
131             if(component>ziel[j]){
                    component=ziel[j];
133                 s=j;
                }
135         }
```

```
137         // Pivotisierung der Zeile
            //-------------------------
139         // Kommt erste Komponente als Vergleichselement in Frage?
            if(matrix[0][s]>0){
141             component=matrix[0][n+m]/(matrix[0][s]);
                z=0;
143         }
            // Existiert eine positive Komponente in der Pivotspalte?
145         else{
                for(int i=0;i<m;i++){
147                 if(matrix[i][s]>0){
                        component=matrix[i][n+m]/(matrix[i][s]);
149                     z=i;
                        h=h+1;
151                 }
                }
153             // Ex esixtiert keine positive Komponente, weswegen die Pivotzeile die
     erste wird, weil erste Komponente definitiv unzulaessig ist.
                if(h==0){
155                 component=matrix[0][n+m]/(matrix[0][s]);
                    z=0;
157             }
            }
159
            // Suche nach kleinstem Quotient
161         for(int i=0;i<m;i++){
                if(component>(matrix[i][n+m]/matrix[i][s]) && matrix[i][s]>0){
163                 component=matrix[i][n+m]/matrix[i][s];
                    z=i;
165             }
            }
167         cout<<endl<<" mit Pivotspalte = "<<(s+1)<<",Pivotzeile = "<<(z+1)<<endl<<
    endl;

169         // Basiswechsel
            pivot=matrix[z][s];
171         if(pivot>0){
                for(int j=0;j<=(n+m);j++){
173                 matrix[z][j]=matrix[z][j]/pivot;
                }
175
                for(int i=0;i<m;i++){
177                 factor=matrix[i][s];
                    if(i!=z){
179                     for(int j=0;j<=(n+m);j++){
                            matrix[i][j]=matrix[i][j]-factor*matrix[z][j];
181                     }
                    }
183             }
```

```
185             factor=ziel[s];
                for(int j=0;j<=(n+m);j++){
187                 ziel[j]=ziel[j]-factor*matrix[z][j];
                }
189             basis[z]=s;


191

                // Koeffizienten der Zielfunktion negativ?
193             for(int j=0;j<(n+m);j++){
                    if(ziel[j]<0){
195                     e=e+1;
                    }}
197             r=r+1;
                cout<<endl<<r<<"-te Simplex-Tableau:"<<endl;
199             printtableau(matrix,basis,ziel);

201             if(e==0){
                    cout<<endl<<endl<<"Die Optimalloesung lautet:"<<endl;
203                 for(int i=0;i<m;i++){
                        if((basis[i]+1)>n){
205                     cout<<"x_"<<(basis[i]+1)<<" = "<<matrix[i][n+m]<<" -->
    Schlupfvariable (beschreibt nicht vollstaendig ausgenuetzte Kapazitaeten),"<<
    endl;
                        }
207                     else{
                            cout<<"x_"<<(basis[i]+1)<<" = "<<matrix[i][n+m]<<", "<<
    endl;}}
209                 cout<<"womit eine optimale, zulaessige Loesung gefunden wurde."<<
    endl;
                }
211         }
            else{
213             r=r+1;
                cout<<endl<<r<<"-te Simplex-Tableau:"<<endl;
215             printtableau(matrix,basis,ziel);
                cout<<endl<<endl<<"Es existiert keine endliche Optimalloesung."<<endl;
217             e=0;
            }
219     }

221   return 0;
}
```

Listing 2: Primal-Simplex-Verfahren

C

Transkript

Ich möchte mich nochmal bei DDr. Philipp Hungerländer für die Möglichkeit bedanken, ein Experteninterview mit ihm zu führen und seine Expertise zu dem Thema Operations Research zu hören, die mir Einblicke in die praktische Welt gab, aber auch das theoretische Konstrukt hinter der mathematischen Optimierung aufzeigte.

Das Interview mit Herrn DDr. Philipp Hungerländer wurde an der Alpen-Adria-Universität in Klagenfurt am 23.07.2018 geführt. Im Transkript wurden die Abkürzungen **B** (Befragter) für die Wortmeldungen von Herrn DDr. Philipp Hungerländer und **I** (Interviewer) für die Wortmeldungen des Autors verwendet.

I: Die erste Frage wäre: Was war bezüglich Optimierung Ihr letztes Projekt, an dem Sie gearbeitet haben?

B: Und das ist für Ihre Arbeit relevant? (*lacht*)

I: Nicht wirklich. Das ist nur eine einsteigende Frage, die mich interessieren würde.

B: Also im Moment, arbeite ich an Projekten mit der Rail Cargo zum Beispiel, über das darf ich reden. Da geht es darum, die Auslastung von den Lokomotiven zu optimieren. Also da hat man eine Menge von Zügen in Österreich, die durchgeführt werden sollten, vorgegeben und dann sucht man eine Zuordnung, also die vorhandenen Lokomotiven so den Zügen zuzuordnen, dass man möglichst wenig Lokomotiven braucht, um alle Züge durchzuführen. Da gibt es eine Menge von Nebenbedingungen, die man beachten muss - wie Wartung der Lokomotive und so weiter, nach gewisser Kilometeranzahl. Und wenn wir es da schaffen, fünf Prozent der Lokomotiven einzusparen, ist das eine hohe Kostenreduktion für die ÖBB, weil jede Lokomotive sehr teuer ist.

I: Und welche Optimierung oder welche Grundlagen werden dabei verwendet, um da zu opti-

mieren? Ist das rein Lineare Optimierung?

B: Lineare Optimierung nicht, das ist eine Ganzzahlige Optimierung, also Ganzzahlige Optimierungsprozesse sind ja grundsätzlich. Man muss Lineare Optimierung verstehen, um Ganzzahlige Optimierungsprobleme verstehen zu können, weil da kommen halt zusätzlich zum LP noch Ganzzahligkeitsbedingungen für die Variablen dazu und das macht das Ganze wesentlich schwieriger, das effizient zu lösen. Deswegen verwenden wir auch - typischerweise in der Praxis - Heuristiken, die auf das Problem speziell zugeschnitten sind, aber das geht dann wahrscheinlich zu weit. Aber die Lineare Optimierung ist auf alle Fälle eine notwendige Grundlage für alles, was wir dort machen. (...) Bei Ganzzahligen Optimierungsprobleme hat man einen Baum, wo man dann die einzelnen Variablen fixiert und in jedem Knoten von dem Baum löst man wieder ein Lineares Programm. Also Lineare Programme sind auch algorithmisch die Grundlage zum Lösen Ganzzahliger Optimierungsprobleme.

I: Also funktioniert die Anwendung Linearer Optimierung auch in komplexeren Situationen?

B: Lineare Optimierung selbst ist zu wenig, aber es ist eine Subroutine für komplexere Algorithmen, die dann in der Praxis eingesetzt werden.

I: Und gibt es dabei außer Simplex-Verfahren andere? Verwenden Sie da auch andere Verfahren zur Lösung?

B: Also es gibt Verfahren: es gibt die Innere-Punkte-Methode, die im Gegensatz zum Simplex-Verfahren auch nachgewiesen polynomielle Laufzeit hat. Das Simplex-Verfahren kann ja immer eine exponentielle Laufzeit haben. Entsprechend ist das theoretisch wichtig, diese Innere-Punkte-Methode, und in den Solvern, die wir haben, kann man dann einfach umstellen, ob man jetzt mit Simplex oder Innere-Punkte-Methode lösen will. Typischerweise ist aber Simplex für die Praxis schneller, überhaupt für die Ganzzahligen Optimierungsprobleme, weil man immer wieder einen Restart machen muss, von ähnlichen Problemen, und da haben diese Aktive-Mengen-Methoden eben bessere Restartfähigkeiten.

I: Aber die unterscheiden sich nur in der Laufzeit und der Herangehensweise?

B: Genau, in der Laufzeit für die jeweilige Probleminstanz unterscheiden sie sich manchmal. Manchmal ist das Simplex-Verfahren schneller, manchmal ist das Innere-Punkte-Verfahren schneller. Aber in der Praxis wird meistens dann, wenn Lineare Optimierung als Subroutine eingesetzt wird, das Simplex-Verfahren verwendet.

I: Welche Computersysteme verwenden Sie, um das zu lösen? Welche Solver?

B: Wir haben einen Server, da an der Uni stehen und da geht man dann mit SSH hin. Da kann man dann auch mehrere Berechnungen gleichzeitig starten, weil er einen entsprechend großen Arbeitsspeicher hat - und Kerne. Selber am Rechner, arbeite ich nur, um zu testen. Wenn ich

dann echtes Benchmarking brauche, mache ich das eigentlich immer am Server.

I: In welcher Programmiersprache ist das?

B: Wenn wir diese Solver verwenden, um sie anzusprechen und die Daten aufzubereiten, verwenden wir typischerweise Java, weil da Kosten-Nutzen am besten ist, sozusagen, weil es wesentlich schneller geht als in C++ oder C und die eigentliche Rechnerei dann im Solver passiert. Wenn man dann diese maßgeschneiderten Heuristiken baut, ist es eine Kombination aus Java und C. Die rechenintensiven Operationen werden dann halt in einer performanteren Programmiersprache umgesetzt. Die weniger rechenintensiven dann in Java, weil es einfach wesentlich schneller programmiert ist.