Prof. Dr. Winfried Hochstättler

Modul 61412

Lineare Optimierung

LESEPROBE

Fakultät für
Mathematik und Informatik

FernUniversität in Hagen

Inhaltsverzeichnis

Kapitel 6

Zur Komplexität des Simplexalgorithmus

Der Simplexalgorithmus hat sich seit seiner Entdeckung als in der Praxis effizientes Verfahren etabliert. Aus Sicht der Theorie hat man dafür keine wirklich befriedigende Erklärung. Wir werden in diesem Kapitel einen ersten, einfacheren Komplexitätsbegriff kennen lernen und den Simplexalgorithmus aus dieser Warte betrachten.

6.1 Streng polynomiale Algorithmen und ein fraktionaler Rucksack

Wenn wir die Güte eines Algorithmus beurteilen wollen, so werden wir auch einem guten Verfahren zugestehen, dass es an größeren Aufgaben länger rechnet. Dafür müssen wir aber zunächst die Größe einer Aufgabe definieren. Außerdem sollten wir zumindest eine ungefähre Definition eines Algorithmus geben.

Unter einem „Algorithmus für ein Problem“ (oder für eine *Problemklasse*) verstehen wir eine Folge von wohldefinierten Regeln bzw. Befehlen, die in einer endlichen Anzahl von Elementarschritten aus jeder spezifischen Eingabe, einer *Instanz* des Problems, eine spezifische Ausgabe erzeugt. Wir forden also:

i) Ein Algorithmus muss sich in einem Text endlicher Länge beschreiben lassen.

ii) Die Abfolge der Schritte ist in jeder Berechnung eindeutig.

iii) Jeder Elementarschritt lässt sich mechanisch und effizient ausführen.

iv) Der Algorithmus stoppt bei jeder Eingabe nach endlich vielen Schritten.

Ein wichtiges Qualitätskriterium eines Algorithmus ist die jeweilige Anzahl der bis zur Terminierung des Algorithmus auszuführenden Elementarschritte. Was ein Elementarschritt ist, hängt vom jeweiligen Maschinenmodell ab. Wir wollen hier nicht auf Details eingehen, sondern auf einschlägige Kurse und Bücher der Komplexitätstheorie, etwa [8, 17, 2] verweisen. Sie können sich unter einem Elementarschritt etwa einen Maschinenbefehl vorstellen. Wir unterscheiden hier zwischen Maschinen, die alle elementaren Rechenoperationen in einem Schritt durchführen können oder – realistischer – solchen, bei denen die Anzahl der Elementarschritte, z.B. für die Multiplikation zweier beliebig großer ganzer Zahlen, auch von deren Größe abhängig ist. Die Anzahl der ausgeführten Elementarschritte wird in beiden Fällen als Laufzeit des Algorithmus bezeichnet.

Werden die Instanzen größer, so wird man dem Algorithmus auch eine längere Laufzeit zugestehen. Deswegen betrachten wir die Laufzeit eines Algorithmus in Abhängigkeit der *Länge* der Eingabedaten. Dabei gehen wir davon aus, dass die Daten in einem sinnvollen Format gespeichert sind. Im Falle einer ganzen Zahl z wählt man als Kodierungslänge üblicherweise $1+\lceil \log_2(1+|z|)\rceil$. Damit kann man das Vorzeichen und den Betrag in der Binärdarstellung von z speichern.

6.1.1 Beispiel. Wenn wir etwa die Zahl -314 binär kodieren wollen und vereinbaren, dass das erste Zeichen als Vorzeichen interpretiert wird, wobei 1 für negatives und 0 für ein positives Vorzeichen steht, so wird $-314 = -(256+32+16+8+2)$ durch die Zeichenkette 1100111010 kodiert, diese besteht aus

$$1+\lceil \log_2(315)\rceil = 1+\lceil 8.2992\rceil = 10$$

Zeichen.

Wir bezeichnen mit $\langle w\rangle$ die *Kodierungslänge* einer Instanz w. Diese Definition der Kodierungslänge wird uns in den folgenden Kapiteln des Kurses begleiten, wenn wir Verfahren kennen lernen werden, die lineare Programme beweisbar effizient lösen. In diesem Kapitel werden wir aber den Simplexalgorithmus im Hinblick auf die *Anzahl $I(w)$ der Daten* in einer Eingabe untersuchen, wobei wir die Länge der Kodierung einzelner Zahlen ignorieren.

6.1.2 Aufgabe. Sei $\max\{c^\top x \mid Ax = b, x \geq 0\}$ eine Instanz w des linearen Optimierungsproblems. Bestimmen Sie grob $\langle w\rangle$ und $I(w)$. Nehmen Sie der Einfachheit halber an, dass alle Eingangsdaten ganzzahlig sind.

Lösung siehe Lösung 6.7.1.

Ist M ein Algorithmus und w die Eingabe, so bezeichnen wir mit $time_M(w)$ die Zahl der Elementarschritte, die M bei Eingabe von w bis zur Terminierung benötigt.

Betrachten wir Maschinenmodelle, bei denen die Multiplikation zweier ganzer Zahlen nicht in einem Elementarschritt möglich ist, so bezeichnen wir mit

$$t_M(n) = \max\{time_M(w) \mid \langle w \rangle = n\}$$

die *Komplexität* oder auch *Worst-Case-Komplexität* des Algorithmus.

Analog definieren wir für Maschinen, bei denen elementare Rechenoperationen Elementarschritte sind

$$t_M^s(n) = \max\{time_M(w) \mid I(w) = n\}.$$

Die genauen Funktionswerte $t_M(n)$ bzw. $t_M^s(n)$ für jedes n zu bestimmen, wird in den seltensten Fällen möglich sein. Die exakten Zahlen sind auch nicht so wesentlich. Interessanter ist das *asymptotische Verhalten*. Dafür haben Sie eventuell im informatischen Nebenfach die „Big-Oh"-Notation kennengelernt. Sollte dies nicht der Fall sein, können Sie sie im Anhang dieser Einheit in Abschnitt 6.6 nachschlagen.

Als sinnvolles Maß für die Effizienz von Algorithmen haben sich polynomiale Schranken für die Laufzeiten erwiesen. Ist $t_M(n) = O(n^k)$ für ein $k \in \mathbb{N}$, so ist die Laufzeit durch ein Polynom in der Kodierungslänge beschränkt. Man spricht von *Polynomialzeit*, und der Algorithmus heißt *polynomial*.

Ist $t_M^s(n) = O(n^k)$ für ein $k \in \mathbb{N}$, so sagen wir, der Algorithmus ist *streng polynomial*.

6.1.3 Aufgabe. Zeigen Sie: Kann man die elementaren Rechenoperationen auf einem Maschinenmodell in Polynomialzeit ausführen und bleibt die Kodierungslänge aller Zahlen bei Zwischenrechnungen polynomial beschränkt, so liefert ein streng polynomialer Algorithmus einen polynomialen Algorithmus.

Lösung siehe Lösung 6.7.2.

Als Beispiel betrachten wir das fraktionale Rucksackproblem:

6.1.4 Beispiel. Seien $c_1, \ldots c_n, a_1, \ldots, a_n, b \in \mathbb{R}_+$. Wir betrachten das lineare Programm

$$\max\left\{\sum_{i=1}^{n} c_i x_i \mid \sum_{i=1}^{n} a_i x_i \leq b, 0 \leq x_i \leq 1\right\}.$$

Verlangt man von den Variablen, dass sie nur die Werte 0 und 1 annehmen dürfen, so haben wir hier das klassische RUCKSACK PROBLEM (engl. KNAPSACK PROBLEM). Dabei interpretiert man b als die Kapazität eines Rucksacks, c_i als den Nutzen des Gegenstandes i und a_i als das Volumen oder Gewicht dieses Gegenstandes. Man möchte nun so Gegenstände in den Rucksack packen, dass die Kapazitätsrestriktion beachtet wird, und der Nutzen maximiert wird.

Ausgehend von einem leeren Rucksack lösen wir das Problem mit der upper bound technique aus Abschnitt 5.12. Als „Pivotregel“ wollen wir dabei vereinbaren, dass wir als Pivotspalte die Nichtbasisvariable nehmen, bei der der Quotient $\frac{c_i}{a_i}$ am größten ist, also mit dem größten Nutzen pro Kapazitätseinheit. Wir nehmen im Folgenden an, dass die Indizes so geordnet sind, dass

$$\frac{c_i}{a_i} \geq \frac{c_{i+1}}{a_{i+1}} \qquad \text{für } 1 \leq i \leq n-1.$$

Wenden wir mit dieser Regel die upper bound technique an, so werden, so lange $\sum_{i=1}^{i_0-1} a_i x_i < b$ ist, nacheinander die Variablen $x_1, \ldots, x_{i_0-1}$ von der unteren auf die obere Schranke gesetzt. Die Basis und die reduzierten Kosten ändern sich nicht, bleiben also auf c. Sobald $\sum_{i=1}^{i_0} a_i x_i \geq b$ ist, nehmen wir i_0 in die Basis auf. Die reduzierten Kosten ändern sich überall zu

$$c_i - a_i \frac{c_{i_0}}{a_{i_0}}.$$

Auf Grund der Annahme an die Sortierung der Indizes ist dieser Ausdruck nun nicht negativ für alle Indizes $i < i_0$. Bei diesen Indizes nehmen die Nichtbasisvariablen den Wert der oberen Schranke an. Ferner ist der Wert nicht positiv für alle Indizes $i > i_0$, bei denen die Variablen auf Null stehen. Also ist das Optimalitätskriterium der upper bound technique erfüllt.

Da die Auswahl jedes einzelnen Pivotelements naiv in $O(n)$ – oder durch Ordnen die gesamte Pivotwahl mit einem Gesamtaufwand von $O(n \log n)$ erledigt werden kann[1], und die Kosten eines Pivotschritts $O(nm)$ betragen, hat dieses Vorgehen eine Laufzeit

$$t_M^s(nm) = O(mn^2 + n \log n),$$

denn nach Aufgabe 6.1.2 ist die Datenmenge $I(w)$ eines linearen Programms gerade $I(w) = O(nm)$. Also ist der Algorithmus streng polynomial.

[1] Für Sortieralgorithmen und deren Komplexität verweisen wir auf [13]

Bleibt hingegen die Ganzzahligkeitsforderung bestehen, so ist das Problem $\mathscr{NP}$-vollständig. Somit ist es vermutlich nicht in Polynomialzeit lösbar. Wir werden im nächsten Kapitel etwas detaillierter über den Begriff der $\mathscr{NP}$-Vollständigkeit sprechen. Für eine exakte Exposition müssen wir aber wieder auf [8, 17, 2] verweisen.

6.2 Personaleinsatzplanung

Wir wollen hier noch ein weiteres Problem der kombinatorischen Optimierung, das gewichtete bipartite Matching, vorstellen, zu dem man einen streng polynomialen Algorithmus kennt, der als Simplexpivotalgorithmus aufgefasst werden kann. Für das allgemeine lineare Optimierungsproblem kennt man einen solchen Algorithmus bis heute, August 2016, noch nicht.

6.2.1 Beispiel. Sie sind in der Personaleinsatzplanung eines Beratungsunternehmens tätig. Da ihre Berater immer zum Kunden geschickt werden, haben sie keine eigenen Büros, sondern fahren von Ihrer eigenen Wohnung zum Kunden. Sie haben n Berater, die Sie bei n Kunden einsetzen wollen, je ein Berater bei einem Kunden. Beim Einsatz von Berater i beim Kunden j entstehen Reisekosten der Höhe $c_{ij} \geq 0$. Ziel ist es, die Berater so einzusetzen, dass die Reisekosten minimiert werden.

Wir modellieren dies als lineares Programm. Die Variable x_{ij} soll angeben, ob Berater i beim Kunden j eingesetzt wird. Dass Berater i genau einmal eingesetzt wird, ergibt die Gleichung

$$\sum_{j=1}^{n} x_{ij} = 1.$$

Dass Kunde j nur einen Berater erhält, ergibt

$$\sum_{i=1}^{n} x_{ij} = 1.$$

Die Variablen x_{ij} sollen 0 oder 1 sein. In der linearen Optimierung können wir nur fordern, dass $0 \leq x_{ij} \leq 1$. In diesem Falle reicht das aber, wie wir gleich zeigen werden. Zunächst einmal geben wir noch als Zielfunktion

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

an.

6.2.2 Definition. Eine Matrix $A \in \mathbb{K}^{m\times n}$ heißt *total unimodular*, wenn alle Unterdeterminanten (Determinanten quadratischer Untermatrizen) einen Wert in $\{0,1,-1\}$ haben.

6.2.3 Proposition. *Sei A total unimodular und $b \in \mathbb{Z}^m$. Ist x^* eine Ecke von*

$$P = \{x \in \mathbb{K}^n \mid Ax \leq b\},$$

so ist $x^* \in \mathbb{Z}^n$ ganzzahlig.

Beweis. Da x^* Ecke ist, ist nach Satz 4.4.2 $\operatorname{rg}(A_{\operatorname{eq}(\{x^*\}).}) = n$. Wir können also $I \subseteq \operatorname{eq}(\{x^*\})$ so wählen, dass $A_{I.}$ eine reguläre Matrix ist und $x^* = A_{I.}^{-1} b_I$. Der Vektor x^* ist somit Lösung des linearen Gleichungssystems $A_{I.}x^* = b_I$. Nach der Cramerschen Regel erhalten wir für $i = 1,\ldots,n$

$$x_i^* = \frac{\det(A_i)}{\det(A_{I.})},$$

wobei die Matrix A_i aus der Matrix $A_{I.}$ entsteht, indem die i-te Spalte durch b ersetzt wird. Da alle Einträge einer Matrix selber (1×1)-Unterdeterminanten sind, hat $A_{I.}$ nur Einträge 0, 1 und -1. Nach Voraussetzung ist $b \in \mathbb{Z}^m$ und damit sind auch alle A_i ganzzahlige Matrizen. Die Determinante einer ganzzahligen Matrix ist aber auch ganzzahlig. Dies liest man etwa sofort an der Leibnizformel ab. Also ist $\det(A_i) \in \mathbb{Z}$. Nach Voraussetzung ist A unimodular. Da $A_{I.}$ regulär ist, schließen wir $\det(A_{I.}) \in \{1,-1\}$. Also ist $x^* \in \mathbb{Z}^n$. □

Um zu zeigen, dass die Matrix unseres Problems total unimodular ist, zeigen wir allgemeiner:

6.2.4 Proposition. *Sei $A \in \mathbb{K}^{m\times n}$ eine total unimodulare Matrix. Dann gilt:*

i) Jede Matrix, die man erhält, wenn man Zeilen oder Spalten von A mit -1 multipliziert, ist total unimodular.

ii) Auch die Matrix $\binom{A}{I_n}$ ist total unimodular.

iii) Sei $B \in \mathbb{K}^{m\times n}$ eine Matrix, die in jeder Spalte höchstens zwei Nichtnulleinträge hat. Diese Nichtnulleinträge von B seien 1 oder -1. Ferner habe B in keiner Spalte zwei gleiche Nichtnulleinträge. Dann ist B total unimodular.

Beweis.

i) Dies folgt sofort aus der Multilinearität der Determinante.

ii) Sei $\tilde{A}$ eine quadratische $(k \times k)$-Untermatrix von $\binom{A}{I_n}$. Wir zeigen die Behauptung per Induktion über k. Da A und I_n Matrizen mit Einträgen aus $\{0,1,-1\}$ sind, ist für $k=1$ die Behauptung richtig. Sei also $k>1$. Ist $\tilde{A}$ quadratische Untermatrix von A, so gilt die Behauptung, da A total unimodular ist. Andernfalls gibt es eine Zeile, die höchstens einen Nichtnulleintrag hat und dieser ist dann gleich 1. Ist die Zeile Null, so gilt dies offensichtlich auch für die Determinante. Andernfalls sei $A_{ij}=1$ der einzige Nichtnulleintrag dieser Zeile. Dann ist

$$\det(\tilde{A}) = (-1)^{i+j}\det(\tilde{A}^{ij}),$$

wobei $\tilde{A}^{ij}$ die $((k-1)\times(k-1))$-Untermatrix von $\tilde{A}$ ist, die durch Streichen der i-ten Zeile und j-ten Spalte entsteht. Die Behauptung folgt dann aus der Induktionsvoraussetzung.

iii) Auch hier führen wir Induktion über die Größe k der quadratischen Untermatrizen und zeigen, dass deren Determinanten jeweils 0, 1 oder -1 sind. Die Matrix B hat nur Einträge 0, 1 und -1. Damit sind alle (1×1)-Unterdeterminanten von diesen Werten und die Induktion ist verankert. Sei also $\tilde{A}$ eine $(k\times k)$-Untermatrix. Wir unterscheiden drei Fälle:

(a) Gibt es eine Spalte, in der nur Nullen stehen, so ist $\det(\tilde{A})=0$.

(b) Gibt es keine Nullspalte, aber eine Spalte mit genau einem Nichtnulleintrag etwa A_{ij}, so entwickeln wir nach dieser Spalte und erhalten

$$\det(\tilde{A}) = (-1)^{i+j-1}A_{ij}\det(\tilde{A}^{ij}),$$

Da $\tilde{A}^{ij}$ auch eine Untermatrix von B ist, hat sie nach Induktionsvoraussetzung die Determinate 0, 1 oder -1. Wegen $A_{ij}\in\{1,-1\}$ gilt dies auch für $\tilde{A}$. Nach dem Induktionsprinzip folgt also die Behauptung.

(c) Im letzten Fall haben wir in jeder Spalte genau eine 1 und eine -1. Dann ist aber die Summe der Zeilen der Nullvektor, also hat die Matrix nicht vollen Rang und $\det(\tilde{A})=0$.

□

Das Polyeder zu unserer Optimierungsaufgabe wird beschrieben durch $Ax=1$, $0\le x\le 1$. Multiplizieren wir dabei in der Matrix A die Zeilen der Kunden mit -1, erhalten wir nach Proposition 6.2.4 iii) eine total unimodulare Matrix, denn jede Variable taucht je genau einmal in einer Kundenzeile und in einer Beraterzeile auf.

Mit A ist nach Proposition 6.2.4 i) auch $\binom{A}{-A}$ total unimodular und schließlich nach Proposition 6.2.4 ii) die Matrix, die den zulässigen Bereich unseres Problems in Ungleichungsform beschreibt. Also sind nach Proposition 6.2.3 alle Ecken unseres linearen Programms ganzzahlig.

Wir wollen unser Problem mit so etwas wie dem dualen Simplexalgorithmus lösen, da wir für das duale Problem sofort eine zulässige Lösung haben. Notieren wir unser primales Programm einmal explizit, so lautet es:

$$\begin{array}{lrcll} & \min \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} & & & \\ \text{unter} & \sum_{i=1}^n x_{ij} & = & 1 & \text{für alle } 1 \leq j \leq n \\ & \sum_{j=1}^n x_{ij} & = & 1 & \text{für alle } 1 \leq i \leq n \\ & x_{ij} & \geq & 0. & \end{array}$$

Das duale Problem lautet dann

$$\begin{array}{lrcll} \max & \sum_{i=1}^n u_i + \sum_{j=1}^n v_j & & & \\ \text{unter} & u_i + v_j & \leq & c_{ij} & \forall 1 \leq i, j \leq n. \end{array} \tag{6.1}$$

Bei unserer Interpretation der folgenden Prozedur als Spezialfall des Simplexalgorithmus haben wir eigentlich das Problem, dass

i) die Variablen nicht vorzeichenbeschränkt sind und

ii) der zulässige Bereich nicht spitz ist.

Addiert man nämlich eine (nicht notwendig positive) Konstante zu allen Beratern und zieht die gleiche Konstante von allen Kunden ab, so erhält diese Operation die Zulässigkeit. Also ist der Linearitätsraum des dualen Problems nicht trivial und somit das zugehörige Polyeder nicht spitz.

Man könnte dies dadurch auflösen, dass man die Variablen aufspaltet. Das würde aber das folgende Verfahren unnötig verkomplizieren.

Wir konstruieren im Folgenden zulässige Lösungen für das duale Problem und dazu komplementäre, nicht notwendig zulässige Lösungen des primalen Problems. Sind wir auch primal zulässig, so haben wir nach dem Satz vom komplementären Schlupf eine Optimallösung gefunden. Wie gesagt, in der Darstellung hier ignorieren wir die Tatsache, dass die duale Lösung keine Ecklösung ist. Wie wir sehen werden, ist die primale Lösung eine Ecke.

Um komplementäre Lösungen zu finden, betrachten wir den *bipartiten Graphen* der *dichten Kanten*.

6.2.5 Definition. Sei $G = (W,E)$ ein Graph. Dann heißt G *bipartit* , wenn es eine Partition $W = U \dot\cup V$ der Knotenmenge gibt, so dass

$$\forall e \in E : e \cap U \neq \emptyset \neq e \cap V,$$

also alle Kanten je einen Endknoten in jeder Menge haben.

Sei (u,v) zulässig für (6.1). Initial setzen wir dafür $(u,v) = (0,0)$. Diese Lösung ist zulässig, da nach Voraussetzung $c_{ij} \geq 0$ ist. Sei U die Menge der Berater und V die Menge der Kunden. Diese Mengen sind disjunkt. Dies sollten Sie im Hinterkopf behalten, auch wenn wir im Folgenden dennoch für beide Mengen die Zahlen von $1,\ldots,n$ verwenden, um nicht unnötig die Notation zu verkomplizieren. Wir definieren dann den bipartiten Graphen G_{uv} der dichten Kanten vermöge

$$W = U \cup V \text{ und } \left(\{i,j\} \in E \iff u_i + v_j = c_{ij}\right).$$

Nun suchen wir im Graph der dichten Kanten nach einer möglichst großen Zuordnung von Beratern zu Kunden. Dies machen wir wie folgt:

- Ausgehend von einer initialen Zuordnung von Beratern zu Kunden versuchen wir, diese zu verbessern.
- Entweder finden wir dabei eine Zuordnung, die mehr Berater-Kunde-Paare enthält, wobei jeder Berater und jeder Kunde, die vorher zugewiesen waren, weiter zugewiesen bleiben, oder
- wir können die Dualvariablen so modifizieren, dass wir eine neue Kante in den Graph der dichten Kanten aufnehmen, die uns den Suchraum vergrößern lässt.

Ausgehend von einem nicht zugeordneten Berater bauen wir dafür einen gerichteten Suchbaum T, einen Wurzelbaum, auf. Zur Klärung, was ein Wurzelbaum ist, machen wir noch ein paar kleine Definitionen.

6.2.6 Definition. Sei $G = (V,E)$ ein Graph. Ein *Zyklus in* G ist eine (geschlossene) Folge von Knoten und Kanten $v_0 e_1 v_1 e_1 v_2 \ldots v_{k-1} e_k v_k$, wobei $e_i = \{v_{i-1}, v_i\}$ für $1 \leq i \leq k$ und $v_0 = v_k$. Ein zyklenfreier Graph heißt *Wald*, ein zusammenhängender Wald heißt *Baum*.

Ein *gerichteter Graph* oder auch *Digraph* ist eine binäre Relation. Ein *Wurzelbaum* ist ein Digraph, dessen zu Grunde liegender Graph (wir „vergessen“ die Richtungen der Kanten) ein Baum ist, bei dem es einen ausgezeichneten Knoten, die Wurzel, gibt und alle Kanten so gerichtet sind, dass man von der Wurzel alle Knoten auf gerichteten Wegen erreichen kann.

Sei also nun eine Zuordnung von Beratern zu Kunden j gegeben und i_0 ein nicht zugeordneter Berater. Wir nehmen dann alle Knoten j mit $\{i_0, j\} \in G_{u,v}$ in unseren Suchbaum T auf, orientieren die Kanten von i_0 nach j und markieren i_0 als abgearbeitet. Gibt es unter den so erreichten j einen Kunden, der noch nicht einem Berater zugeordnet ist, so können wir die Zuordnung vergrößern. Andernfalls nehmen wir die zugeordneten Berater in T auf, wobei die Kanten vom Kunden zum Berater orientiert werden. So lange wir nun noch einen nicht abgearbeiteten Berater in T haben, untersuchen wir, ob wir von ihm aus neue Kunden in T aufnehmen können. Finden wir dabei einen Kunden j_0, der noch keinen Berater hat, so besteht der Weg von i_0 nach j_0 in T abwechselnd aus Kanten, die nicht einer Zuordnung entsprechen und Kanten, die einer Zuordnung entsprechen. Lassen wir die Kanten auf dem Weg die Rollen wechseln, erhalten wir eine Zuordnung, bei der zusätzlich i_0 nicht mehr untätig ist und j_0 beraten wird. Andernfalls nehmen wir die Berater der neu gefundenen Knoten wie eben in T auf.

Endet die Suche, ohne dass wir die Zuordnung vergrößern konnten, so sind alle Kunden in T mit einem Berater versorgt. Da auch i_0 in T ist, enthält T einen Berater mehr, als er Kunden enthält. Erhöhen wir nun die u-Variablen der Berater in T und erniedrigen die v-Variablen der Kunden in T, so bleiben alle Kanten von T im Graph der dichten Kanten erhalten, und wir verbessern die Zielfunktion. Genauer ändern wir den Wert aller dieser Dualvariablen um

$$\min\{c_{ij} - u_i - v_j \mid i \in T, j \notin T\} > 0.$$

Wird dieses Minimum in der Kante $\{i, j\}$ angenommen, so gehört diese zum Graph der dichten Kanten bzgl. der aufdatierten Dualvariablen und wir können die Suche fortsetzen.

Das Verfahren endet, wenn alle Berater mit Kunden versorgt sind. Dies liefert uns eine Lösung des Ausgangsproblems. Da alle Variablen x_{ij} auf 0 oder 1 stehen, ist das eine Ecklösung, denn sie kann nicht echte Konvexkombination von Punkten des zulässigen Bereichs sein.

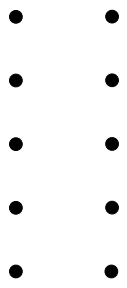
- Wir wollen nun grob die Komplexität dieses Verfahrens abschätzen. Ist w eine Instanz des Problems, so ist $I(w) = n^2 + n = O(n^2)$, da die Daten aus den paarweisen Entfernungen zwischen Beratern und Kunden bestehen.
- Die Menge der eingesetzten Berater können wir höchstens n mal vergrößern.
- Beim Hinzufügen einer Kante zum Suchbaum wächst die Anzahl der Knoten im Suchbaum. Das kann höchstens $2n-1$ mal passieren.

- Das Bestimmen von $\min\{c_{ij} - u_i - v_j \mid i \in T, j \notin T\} > 0$ kann naiv in $O(n^2)$ erledigt werden.
- Also haben wir für $I(w) = n^2$ eine Gesamtlaufzeit von $O(n^4) = O((n^2)^2)$.

Unser primal-duales Verfahren ist also streng polynomial mit quadratischer Laufzeit.

Und nun noch ein Rechenbeispiel im Beispiel. Wir betrachten ein Problem mit 5 Beratern und 5 Kunden. Die Einsatzkosten finden wir in der folgenden Matrix wieder, wobei die Zeilen den Beratern und die Spalten den Kunden zugeordnet sind. Die Dualvariablen tragen wir links neben den Zeilen bzw. über den Spalten ab. Da alle Daten nicht-negativ sind, können wir zu Beginn alle Dualvariablen auf Null setzen.

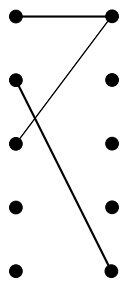
$$\begin{array}{c|ccccc} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 6 & 7 & 8 & 7 & 2 \\ 0 & 1 & 3 & 4 & 4 & 5 \\ 0 & 3 & 6 & 2 & 8 & 7 \\ 0 & 4 & 1 & 3 & 5 & 4 \end{array}$$



Die Dualvariablen sind genau dann zulässig, wenn die Summen aus Zeilen- und Spaltenvariable stets kleiner oder gleich dem Eintrag in der Matrix sind. Dichte Kanten erhalten wir, wo Gleichheit herrscht. Zu Anfang ist also der Graph der dichten Kanten leer und die Suche, welche beim ersten Berater startet, endet auch direkt wieder. Folglich erhöhen wir die Variable des ersten Beraters auf 1 und ordnen ihn dem ersten Kunden zu. Im nächsten Schritt setzen wir die Dualvariable des zweiten Beraters auf 2 und ordnen ihn dem fünften Kunden zu. Im dritten Schritt erhöhen wir die Dualvariable des dritten Beraters auf 1, können aber den Berater nicht dem ersten Kunden zuordnen, da dieser bereits vom ersten Berater versorgt wird.

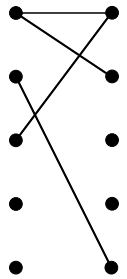
Wir haben also nun folgende Situation:

$$\begin{array}{c|ccccc} & 0 & 0 & 0 & 0 & 0 \\ 1 & \boxed{\mathbf{1}} & 2 & 3 & 4 & 5 \\ 2 & 6 & 7 & 8 & 7 & \boxed{\mathbf{2}} \\ 1 & \boxed{1} & 3 & 4 & 4 & 5 \\ 0 & 3 & 6 & 2 & 8 & 7 \\ 0 & 4 & 1 & 3 & 5 & 4 \end{array}$$



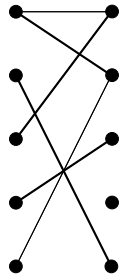
Wenn wir ausgehend vom dritten Berater suchen, haben wir Berater 1 und 3, sowie Kunden 1 in T. Also erhöhen wir die Variablen von Berater 1 und 3 und erniedrigen die von Kunde 1. Da $c_{12} = 2$ ist, können wir nur um 1 erhöhen, ohne Zulässigkeit zu verlieren.

$$\begin{array}{c} \\ 2 \\ 2 \\ 2 \\ 0 \\ 0 \end{array} \begin{array}{c} \begin{array}{ccccc} -1 & 0 & 0 & 0 & 0 \end{array} \\ \begin{pmatrix} \boxed{1} & \boxed{\mathbf{2}} & 3 & 4 & 5 \\ 6 & 7 & 8 & 7 & \boxed{\mathbf{2}} \\ \boxed{\mathbf{1}} & 3 & 4 & 4 & 5 \\ 3 & 6 & 2 & 8 & 7 \\ 4 & 1 & 3 & 5 & 4 \end{pmatrix} \end{array}$$



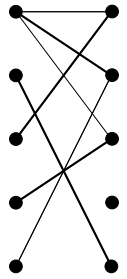
Wir ordnen nun Berater 1 Kunden 2 und Berater 3 Kunden 1 zu. Im nächsten Schritt erhöhen wir die Dualvariable von Berater 4 auf zwei und ordnen ihn dem dritten Kunden zu. Dann setzen wir die Dualvariable von Berater 5 auf 1. Allerdings ist Kunde 2 schon versorgt. Also konstruieren wir wieder unseren Baum T.

$$\begin{array}{c} \\ 2 \\ 2 \\ 2 \\ 2 \\ 1 \end{array} \begin{array}{c} \begin{array}{ccccc} -1 & 0 & 0 & 0 & 0 \end{array} \\ \begin{pmatrix} \boxed{1} & \boxed{\mathbf{2}} & 3 & 4 & 5 \\ 6 & 7 & 8 & 7 & \boxed{\mathbf{2}} \\ \boxed{\mathbf{1}} & 3 & 4 & 4 & 5 \\ 3 & 6 & \boxed{\mathbf{2}} & 8 & 7 \\ 4 & \boxed{1} & 3 & 5 & 4 \end{pmatrix} \end{array}$$



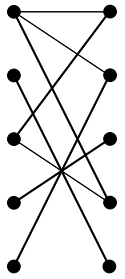
Dieser enthält die Berater 1,3 und 5 und die Kunden 2 und 1. Wegen $c_{13} = 3$ erhöhen wir wieder nur um 1.

$$\begin{array}{c} \\ 3 \\ 2 \\ 3 \\ 2 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccccc} -2 & -1 & 0 & 0 & 0 \end{array} \\ \begin{pmatrix} \boxed{1} & \boxed{\mathbf{2}} & \boxed{3} & 4 & 5 \\ 6 & 7 & 8 & 7 & \boxed{\mathbf{2}} \\ \boxed{\mathbf{1}} & 3 & 4 & 4 & 5 \\ 3 & 6 & \boxed{\mathbf{2}} & 8 & 7 \\ 4 & \boxed{1} & 3 & 5 & 4 \end{pmatrix} \end{array}$$



Durch die neue Kante gelangen auch Berater 4 und Kunde 3 in den Baum. Wegen $c_{14} = c_{34} = 4$ ändern wir die Dualvariablen wieder nur um 1.

$$\begin{array}{c} \\ 4 \\ 2 \\ 4 \\ 3 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccccc} -3 & -2 & -1 & 0 & 0 \end{array} \\ \begin{pmatrix} \boxed{1} & \boxed{2} & \boxed{3} & \boxed{\mathbf{4}} & 5 \\ 6 & 7 & 8 & 7 & \boxed{\mathbf{2}} \\ \boxed{\mathbf{1}} & 3 & 4 & \boxed{4} & 5 \\ 3 & 6 & \boxed{\mathbf{2}} & 8 & 7 \\ 4 & \boxed{\mathbf{1}} & 3 & 5 & 4 \end{pmatrix} \end{array}$$



Nun bleibt Berater 5 dem Kunden 2 zugeordnet, Berater 1 berät nun Kunde 4, Berater 3 Kunde 1. Berater 2 ist weiter für Kunde 5 und Berater 4 weiterhin für Kunde 3 zuständig. Die Gesamtkosten sind mit 10 minimal.

6.2.7 Aufgabe. Betrachten Sie den etwa aus dem Kurs „Lineare Algebra“ bekannten *euklidischen Algorithmus* zur Bestimmung des größten gemeinsamen Teilers zweier natürlicher Zahlen.

Zeigen Sie: Der euklidische Algorithmus ist ein polynomiales, aber kein streng polynomiales Verfahren.

Lösung siehe Lösung 6.7.3.

6.3 Klee-Minty Cubes

Wir hatten im letzten Abschnitt angedeutet, dass kein streng polynomiales Verfahren für das Problem der linearen Optimierung bekannt ist. Da andererseits ein einzelner Pivotschritt in beiden Modellen nur polynomial viel Rechenzeit benötigt, würde eine Pivotregel für den Simplexalgorithmus, die einen polynomialen Algorithmus liefert, schon eine streng polynomiale Variante des Simplexalgorithmus liefern. Leider ist keine polynomiale Pivotregel bekannt. Im Gegenteil. Erst kürzlich sind zu einigen Regeln, deren „Scheitern“ man vorher noch nicht explizit beweisen konnte, Klassen von linearen Programmen gefunden worden, die zeigen, dass die Regeln nicht polynomial sind. Wir werden auf die jüngere Literatur am Ende dieses Abschnitts noch einmal hinweisen.

Zunächst wollen wir explizit ein klassisches Beispiel angeben, bei dem der Simplexalgorithmus unter Verwendung der Dantzigschen Regel exponentiell viele Iterationen benötigt. Die Darstellung orientiert sich dabei an [21].

Die zu Grunde liegende Idee ist die Folgende. Der Hyperwürfel

$$H_n := \{x \in \mathbb{R}^n \mid 0 \leq x_i \leq 1\}$$

ist durch nur n Ungleichungen und n Vorzeichenrestriktionen gegeben. Die Inputgröße ist also $I(w) = n^2$. H_n hat aber 2^n Ecken, nämlich alle Vektoren in $\{0,1\}^n$. Wenn wir nun den Simplexalgorithmus durch eine geeignete Verzerrung dieses Polyeders dazu bringen können, bei Vorgehen mit der Dantzigschen Regel alle Ecken des so verzerrten Hyperwürfels zu durchlaufen, dann zeigt dies, dass das Verfahren exponentielle Laufzeit in der Anzahl der Variablen hat. Dies kann nicht polynomial in der Inputdatengröße sein.

Die folgende Klasse linearer Programme (LP_n) für $n \in \mathbb{N}$ leistet das Gewünschte, wie wir nun zeigen werden.

$$
(LP_n) \qquad \begin{array}{lrcrcrcrcrcr}
\max & 2^{n-1}x_1 & + & 2^{n-2}x_2 & + & \ldots & + & 2x_{n-1} & + & x_n & & \\
\text{unter} & x_1 & & & & & & & & & \leq & 5 \\
& 4x_1 & + & x_2 & & & & & & & \leq & 25 \\
& 8x_1 & + & 4x_2 & + & x_3 & & & & & \leq & 125 \\
& \vdots & & & & & & & & & & \vdots \\
& \vdots & & & & & & & & & & \vdots \\
& 2^n x_1 & + & 2^{n-1}x_2 & + & \ldots & + & 4x_{n-1} & + & x_n & \leq & 5^n \\
& & & & & & & & & x & \geq & 0
\end{array}
$$

Zunächst einmal wollen wir nachweisen, dass es sich hierbei tatsächlich um einen verzerrten Hyperwürfel handelt.

Wir beobachten:

6.3.1 Proposition. *Sei x zulässig für (LP_n). Dann gilt für alle $i = 1, \ldots, n$:*

i) Ist $x_i = 0$, so ist die i-te Ungleichung strikt.

ii) Ist die i-te Ungleichung nicht strikt, so ist $x_i > 0$.

Beweis.

i) Die Behauptung ist offensichtlich richtig für $i = 1$. Ist $i > 1$, so liefert die $(i-1)$-te Ungleichung

$$2^{i-1}x_1 + 2^{i-2}x_2 + \ldots + 4x_{i-2} + x_{i-1} \leq 5^{i-1}.$$

Hieraus erhalten wir

$$\begin{aligned} & 2^i x_1 + 2^{i-1} x_2 + \ldots + 4x_{i-1} + x_i \\ \overset{x_i=0}{=} \quad & 2(2^{i-1} x_1 + \ldots + 4x_{i-2} + x_{i-1}) + 2x_{i-1} \\ \overset{x_{i-1} \leq 5^{i-1}}{\leq} \quad & 2(\underbrace{2^{i-1} x_1 + \ldots + 4x_{i-2} + x_{i-1}}_{\leq 5^{i-1}}) + 2 \cdot 5^{i-1} \\ \leq \quad & 4 \cdot 5^{i-1} < 5^i. \end{aligned}$$

Also ist die i-te Ungleichung strikt.

ii) Da nach Voraussetzung $x_i \geq 0$ ist, handelt es sich bei ii) um die Kontraposition von i).

□

Nach Proposition 6.3.1 hat (LP_n) also keine entarteten Ecken und jede Ecke besteht aus einem komplementären Satz von Ungleichungs- und Vorzeichenrestriktionen. Dass alle diese komplementären Paare auch eine Ecke liefern, besagt der folgende Satz:

6.3.2 Satz. *Der zulässige Bereich von (LP_n) hat 2^n Ecken. Und zwar erhält man für jede Teilmenge $S \subseteq \{1, \ldots, n\}$ eine Ecke, wenn man $x_i = 0$ für alle $i \in S$ setzt und Gleichheit der Ungleichungsnebenbedingungen in allen übrigen Indizes fordert.*

Beweis. Dass wir keine weiteren Ecken erhalten, folgt aus Proposition 6.3.1. Auf Grund der unteren Dreiecksgestalt der Restriktionsmatrix führen alle diese Wahlen der Nichtnegativitäts- und Ungleichungsrestriktionen zu regulären Gleichheitsmatrizen, liefern also Basen. Offensichtlich sind zugehörigen Ecken paarweise verschieden. Wir müssen also nur noch nachrechnen, dass sie zulässig sind. Somit müssen wir für die zugehörigen Ecklösungen nachweisen, dass die von Null verschiedenen Koordinaten nicht negativ sind, und dass die nicht mit Gleichheit erfüllten Ungleichungen nicht verletzt werden. Dies zeigen wir mittels vollständiger Induktion über den Index i.

Für $i = 1$ ist entweder $x_1 = 0$ oder $x_1 = 5$. In beiden Fällen sind die Behauptungen erfüllt.

Sei also $i > 1$. Falls $x_i = 0$ ist, so haben wir eben berechnet, dass

$$\begin{aligned} 2^i x_1 + 2^{i-1} x_2 + \ldots + 4x_{i-1} + x_i \quad & \overset{x_i=0}{=} \quad 2(\underbrace{2^{i-1} x_1 + \ldots + 4x_{i-2} + x_{i-1}}_{\overset{IV}{\leq} 5^{i-1}}) \\ & \leq \quad 4 \cdot 5^{i-1} < 5^i. \end{aligned}$$

Also ist die i-te Ungleichung strikt erfüllt. Ebenso hatten wir gefunden, dass

$$2^i x_1 + 2^{i-1} x_2 + \ldots + 4x_{i-1} + x_i = 5^i$$

nach Induktionsvoraussetzung die Ungleichung

$$x_i \quad \geq 5^i - 4 \cdot 5^{i-1} > 0$$

impliziert. □

6.3.3 Korollar. *i) Sei I eine Teilmenge der Ungleichungen und $J+n$ eine Teilmenge der Vorzeichenrestriktionen, wobei wir „$x_i \geq 0$“ mit dem Index $i+n$ versehen haben. Dann ist*

$$\text{face}(I \cup (J+n)) \neq \emptyset \iff I \cap J = \emptyset.$$

ii) Der Seitenflächenverband des zulässigen Bereichs ist isomorph zum Seitenflächenverband von H_n.

Beweis. Als Übung. □

6.3.4 Aufgabe. Beweisen Sie Korollar 6.3.3.

Lösung siehe Lösung 6.7.4.

Bevor wir den induktiven Aufbau von (LP_n) nutzen, um nachzuweisen, dass der Simplexalgorithmus mit Dantzigscher Pivotregel, ausgehend von der Startecke $(0,\ldots,0)^\top$, tatsächlich alle Ecken durchläuft, geben wir zunächst einmal an, wo das Optimum angenommen wird.

6.3.5 Proposition. *Das eindeutige Optimum von (LP_n) wird in $(0,\ldots,0,5^n)^\top$ angenommen.*

Beweis. LP_n ist vom Typ

$$\{\max c^\top x \mid Ax \leq b, x \geq 0\}.$$

Als duales Problem erhalten wir also nach Lösung 3.6.5 v)

$$\{\min y^\top b \mid y^\top A \geq c^\top, y \geq 0\}.$$

Ausführlich haben wir also:

$$\begin{array}{rrcrcccrcrcl}
\min & 5y_1 & + & 25y_2 & + & \ldots & + & 5^{n-1}y_{n-1} & + & 5^n y_n & & \\
\text{unter} & y_1 & + & 4y_2 & + & 8y_3 & + & \ldots & + & 2^n y_n & \geq & 2^{n-1} \\
 & & & y_2 & + & 4y_3 & + & \ldots & + & 2^{n-1}y_n & \geq & 2^{n-2} \\
 & & & & & y_3 & + & \ldots & + & 2^{n-2}y_n & \geq & 2^{n-3} \\
 & & & & & & \ddots & & & & & \vdots \\
 & & & & & & & \ddots & & & & \vdots \\
 & & & & & & & y_{n-1} & + & 4y_n & \geq & 2 \\
 & & & & & & & & & y_n & \geq & 1 \\
 & & & & & & & & & y_1,\ldots,y_n & \geq & 0.
\end{array}$$

Die Belegung $y_n = 1, y_i = 0$ für $i = 0,\ldots,n-1$ ist zulässig für das duale Optimierungsproblem mit Zielfunktionswert 5^n. Da das primale Optimierungsproblem mit der angegebenen Lösung den gleichen Zielfunktionswert annimmt, müssen beide nach dem Dualitätssatz Optimallösungen sein.

Ist andererseits x^* eine Optimallösung der primalen Aufgabenstellung, so haben wir wegen der Zielfunktion und der letzten Restriktion

$$\begin{array}{rcl}
2^{n-1}x_1^* + 2^{n-2}x_2^* + \ldots + 2x_{n-1}^* + x_n^* & = & 5^n \\
2^n x_1^* + 2^{n-1}x_2^* + \ldots + 4x_{n-1}^* + x_n^* & \leq & 5^n
\end{array}$$

und somit

$$2^{n-1}x_1^* + 2^{n-2}x_2^* + \ldots + 2x_{n-1}^* \leq 0.$$

Da alle Variablen nicht-negativ sind, folgt hieraus $x_1^* = \ldots = x_{n-1}^* = 0$ und somit $x_n^* = 5^n$, also auch die Eindeutigkeit der Lösung.

□

Nach diesen Vorbereitungen zeigen wir:

6.3.6 Satz. *Der Simplex-Algorithmus mit der Dantzigschen Pivotregel benötigt für* (LP_n) $2^n - 1$ *Pivotsschritte.*

Beweis.

Die Behauptung ist offensichlich äquivalent dazu, dass wir bei diesem Verfahren 2^n Tableaus durchlaufen. Dies zeigen wir mittels vollständiger Induktion über n. Genauer zeigen wir:

i) (LP_n) durchläuft bei Anwendung der Dantzigschen Gradientenregel 2^n Tableaus.

ii) In jedem der Tableaus sind die reduzierten Kosten ganzzahlig.

iii) Die Nicht-Null-Einträge in den reduzierten Kosten sind paarweise verschieden.

Für $n = 1$ ist dies trivial, da wir in einem Schritt von 0 auf 5 wechseln und die reduzierten Kosten von $(1,0)$ auf $(0,-1)$ wechseln. Sei $n > 1$. Nach Induktionsannahme haben das erste und das 2^{n-1}-te Tableau von LP_{n-1} folgende Gestalt

c	1	0	0	0
A	0	I_{n-2}	0	b
$2c$	1	0	1	5^{n-1}

Tableau 1

$-c$	0	0	-1	-5^{n-1}
A	0	I_{n-2}	0	b
$2c$	1	0	1	5^{n-1}

Tableau 2^{n-1}

Dabei haben wir die $(n-1)$-ste Variable und die $(n-1)$-ste Nebenbedingung mit ihrer Schlupfvariablen abgespalten.

Betrachten wir nun zunächst das Starttableau für das Problem (LP_n).

$2c$	2	1	0	0	0	0
A	0	0	I_{n-2}	0	0	b
$2c$	1	0	0	1	0	5^{n-1}
$4c$	4	1	0	0	1	5^n

Tableau 1

Streichen wir aus Tableau 1 die Spalten der n-ten Variable und der n-ten Schlupfvariable sowie die n-te Gleichungszeile, so erhalten wir das Startableau von LP_{n-1}, bei dem die Zeile der reduzierten Kosten mit zwei durchmultipliziert wurde. Nach Induktionsvoraussetzung sind aber die reduzierten Kosten von (LP_{n-1}) während des Durchlaufs des Algorithmus zu jeder Zeit ganzzahlig. Also sind sie hier stets Vielfache von zwei. Insbesondere wird die Spalte der n-ten Variable nach der Dantzigschen Pivotregel nicht zur Pivotspalte, solange wir wie in LP_{n-1}, also auch nicht auf einem Element der letzten Zeile, pivotieren. Dies ist aber dadurch sicher gestellt, dass nach Satz 6.3.2 die n-te Schlupfvariable die Basis nur dann verlassen darf, wenn die n-te Variable in die Basis aufgenommen wird. Also durchlaufen wir alle Schritte von (LP_{n-1}), bis dieses Unterproblem optimal ist. Nach Induktionsvoraussetzung geschieht das im 2^{n-1}-sten Tableau, das also wie folgt aussieht:

$-2c$	0	1	0	-2	0	$-2\cdot 5^{n-1}$
A	0	0	I_{n-2}	0	0	b
$2c$	1	0	0	1	0	5^{n-1}
$-4c$	0	1	0	-4	1	5^{n-1}

Tableau 2^{n-1}

Als einzige Spalte mit positiven reduzierten Kosten bleibt nur die Spalte der n-ten Variablen und wir erhalten als nächstes Tableau:

$2c$	0	0	0	2	-1	$-3\cdot 5^{n-1}$
A	0	0	I_{n-2}	0	0	b
$2c$	1	0	0	1	0	5^{n-1}
$-4c$	0	1	0	-4	1	5^{n-1}

Tableau $2^{n-1}+1$

Auch dieses sieht wieder aus wie ein erweitertes erstes Tableau für LP_{n-1}, nur dass zusätzlich die Spalten der $(n-1)$-ten Variablen und der $(n-1)$-sten Schlupfvariablen vertauscht sind. Dies ändert jedoch nichts an der Pivotwahl, die diese Vertauschung nachvollziehen muss, da die Nichtnulleinträge in den reduzierten Kosten paarweise verschieden sind und deswegen die Spalte mit größtem Eintrag stets eindeutig ist. Folglich müssen wir wiederum 2^{n-1} Iterationen machen, um bei Tableau 2^n zu landen.

$-2c$	-2	0	0	0	-1	-5^n
A	0	0	I_{n-2}	0	0	b
$2c$	1	0	0	1	0	5^{n-1}
$4c$	4	1	0	0	1	5^n

Tableau 2^n

Dieses ist optimal, wurde aber erst nach 2^n-1 Schritten erreicht. □

Wir haben somit bewiesen:

6.3.7 Satz. *Der Simplexalgorithmus unter Verwendung der Dantzigschen Pivotregel ist kein streng polynomialer Algorithmus und also auch kein polynomialer Algorithmus.*

Beweis. Die Inputdatengröße ist $I(LP_n) = O(n^2)$ bzw. $\langle LP_n\rangle = O(n^3)$, da die größte vorkommende Zahl 5^n unter Verwendung von höchstens $3n$ Bits kodiert werden kann.

Also ist, unabhängig von den Kosten der Rechenoperationen, in beiden Modellen schon die Anzahl der Pivotschritte exponentiell in der Anzahl der Variablen. □

6.3.8 Bemerkung. Dieses Beispiel wurde für verschiedene Pivotregeln modifiziert. So läßt sich zeigen, dass z.B. auch Bland's Rule, die größter-Fortschritt-Regel und die steilster-Anstieg-Regel keine polynomialen Pivotregeln sind. Ersteres wird durch leichte Modifikation der Argumente für die Dantzigsche Pivotregel im Prinzip mit dem gleichen Beispiel erreicht. Für die anderen beiden Regeln sehen die Beispiele anders aus.

Bis vor kurzem gab es einige wenige Pivotregeln, bei denen keine Beispiele bekannt waren, die zeigten, dass diese nicht zu polynomialen Algorithmen führten. Die beiden prominentesten dabei sind Zadehs Least-Entered-Rule und Cunninghams Round-Robin-Rule:

Least-Entered-Rule: Wähle als Pivotspalte diejenige, die bisher am wenigsten häufig in die Basis aufgenommen worden ist. Die Zeilenauswahlregel ist nicht genau spezifiziert.

Round-Robin-Rule: Fixiere eine Ordnung der Variablen. Wähle als Pivotspalte diejenige, deren Variable am längsten nicht angefasst worden ist. Wähle als Pivotzeile diejenige, deren basisverlassende Variable am längsten in der Basis war.

Oliver Friedmann hat in seiner Dissertation [7] unter anderem für diese Regeln gezeigt, dass sie mehr als polynomial viele Pivotschritte benötigen können. Dafür hat er 2012 den Tucker Prize der Mathematical Optimization Society erhalten.

6.3.9 Aufgabe. Zeigen Sie: Der Simplexalgorithmus unter Verwendung von Blands Rule ist kein Polynomialzeitalgorithmus.

Lösung siehe Lösung 6.7.5.

Zu folgender Vermutung von W.M. Hirsch wurde erst 2010 ein Gegenbeispiel gefunden [19]. Wie Sie sehen, hat es auch in diesem prominenten Fall 2 Jahre bis zur Veröffentlichung der Arbeit gedauert.

Hirsch-Vermutung 1957, falsifiziert 2010 von Santos: Sei P ein d-dimensionales Polytop mit k Facetten und seien v,w Ecken von P. Dann gibt es im 1-Skelett von P einen Pfad von v nach w der Länge höchstens $k-d$.

Der Durchmesser des 1-Skeletts könnte trotz der Gegenbeispiele von Santos noch immer linearen Durchmesser haben, oder Durchmesser polynomial in m und n. Könnte man solche kurzen Pfade effizient durch lokale Auswahl an jeder Ecke bestimmen, so hätte man eine polynomiale Pivotregel gefunden.

Andererseits würde eine untere Schranke, die beweist, dass der Durchmesser einer Klasse von Polytopen nicht polynomial beschränkt ist, zeigen, dass der Simplexalgorithmus mit keiner Pivotregel ein polynomiales Verfahren werden kann.

6.4 Die mittlere Laufzeit des Simplexalgorithmus

In diesem Abschnitt werden wir ein Resultat vorstellen, das beweist, dass, bei Verwendung der so genannten *Schatteneckenregel* und unter gewissen Annahmen an die Verteilung der linearen Optimierungsprobleme, die erwartete Anzahl an Iterationen des Simplexverfahrens im Gegensatz zum worst-case sogar linear ist. Dieses Resultat, mit dem eine Beobachtung, die man in der Praxis heuristisch schon gemacht hatte, theoretisch untermauert zu werden schien, wurde 1982 von Borgwardt erzielt. Wir geben hier ein verbessertes Resultat von Haimovich wieder. In der Darstellung orientieren wir uns auch hier an [21].

Wir betrachten hier zunächst lineare Optimierungsaufgaben in der Form $\max c^\top x$ unter $Ax \leq b \in \mathbb{R}^m$ und werden danach weitere Resultate für den allgemeinen Fall nur zitieren.

Zunächst einmal wollen wir die Schatteneckenregel vorstellen. Diese wird lokal in der jeweiligen Ecke x_0 von $P(A,b)$ definiert. Dafür wählen wir zunächst einen zufälligen Vektor $\bar{c} \in P(A_{\mathrm{eq}(x_0).}, 0)^\Delta$.

6.4.1 Proposition. *x_0 ist Optimallösung des linearen Programms $\max \bar{c}^\top x$ unter $Ax \leq b$ genau dann, wenn $\bar{c} \in P(A_{\mathrm{eq}(x_0).}, 0)^\Delta$.*

Beweis. Nach Lemma 4.9.3 ist

$$P(A_{\mathrm{eq}(x_0).}, 0)^\Delta = \mathrm{Cone}(A^\top_{\mathrm{eq}(x_0).}).$$

Ist also $\bar{c} \in P(A_{\mathrm{eq}(x_0).}, 0)^\Delta$, so gibt es ein $u_0 \geq 0$ mit $u_0^\top A = \bar{c}$, welches komplementär zu x_0 bzgl. $Ax \leq b$ ist. Das duale Problem zu $\max \bar{c}^\top x$ unter $Ax \leq b$ lautet aber

$$\min u^\top b \text{ unter } u^\top A = \bar{c} \text{ und } u \geq 0.$$

Also ist u_0 zulässig für das duale Problem und komplementär zu x_0. Die Behauptung folgt damit aus dem Satz vom komplementären Schlupf 3.5.1.

Ist x_0 umgekehrt Optimallösung des linearen Programms

$$\max \bar{c}^\top x \text{ unter } Ax \leq b,$$

so gibt es nach dem Satz vom komplementären Schlupf ein u, welches zulässig für das duale Problem und komplementär zu x_0 ist. Also ist $\bar{c} \in P(A_{\text{eq}(x_0)\cdot}, 0)^\Delta$. □

Das lineare Funktional $\bar{c}^\top x$ nennen wir *Kozielfunktion*.

Die Idee ist nun, das Polyeder auf die von $\bar{c}$ und c aufgespannte Ebene zu projizieren, die verbessernde Nachbarecke in dieser Projektion zu bestimmen und dann eine Nachbarecke von x_0 im Ausgangspolyeder zu suchen, die auf diese Ecke projeziert wurde. Daraufhin wird dann die Kozielfunktion so an die neue Ecke angepasst, dass Ziel- und Kozielfunktion weiterhin die gleiche Ebene aufspannen und die neue Ecke optimal bzgl. der Kozielfunktion ist.

Die *Schatteneckenregel* von Karl Heinz Borgwardt lautet dann:

Schatteneckenregel: In der zulässigen Ecke x_k, die $(\lambda c + \bar{c})^\top x$ für ein $\lambda \geq 0$, aber nicht $c^\top x$ über P maximiert, wähle einen Nachbarn x_{k+1} von x_k, der $(\lambda' c + \bar{c})^\top x$ für ein $\lambda' > \lambda$ maximiert oder bestimme eine Extremale μy mit Ecke x_k und $(\lambda' c + \bar{c})^\top y > 0$ für ein $\lambda' > \lambda$.

Den Simplexalgorithmus unter Verwendung der Schatteneckenregel wollen wir ab nun als *Schatteneckenalgorithmus* bezeichnen. Ausgehend von x_0 und $\lambda = 0$ ist dies eine zulässige Pivotregel. Sie ist auch leicht zu implementieren. Beim Übergang von λ zu λ' wird zu einem Zeitpunkt die gesamte Kante von x_k und x_{k+1} optimal sein. Die reduzierten Kosten von $\lambda c + \bar{c}$ lassen sich durch Addition von einem Vielfachen der reduzierten Kosten von c aufdatieren. Also wählen wir ein μ so, dass das Optimalitätskriterium für x_k erhalten bleibt, aber zusätzlich die reduzierten Kosten für ein Nichtbasiselement j Null werden. Dieses pivotieren wir in die Basis und erhalten so x_{k+1} und setzen $\lambda' = \lambda + \mu$, wobei wir in Gedanken in der Kozielfunktion das λ' noch einen Ticken größer machen. Für das Rechnen ist das aber eher unbequem.

Bevor wir ein Beispiel durchrechnen, wollen wir aber zunächst das theoretische Resultat herleiten. Wir hatten angekündigt, dass wir dafür gewisse Annahmen an die Wahrscheinlichkeitsverteilung der linearen Optimierungsprobleme machen. Unsere Grundmenge besteht dabei für feste Parameter $m, n \in \mathbb{N}$ aus allen Quadrupeln $(A, b, c, \bar{c})$ mit $A \in \mathbb{K}^{m \times n}, b \in \mathbb{K}^m, c, \bar{c} \in \mathbb{K}^n$. Die Annahmen sehen zunächst einmal recht harmlos und plausibel aus:

i) Für feste Kozielfunktion ändert die Invertierung eines Ungleichheitszeichens (genauer die Multiplikation einer Zeile mit -1) in $Ax \leq b$, falls die Lösung über dem modifizierten System weiterhin endlich ist, die Wahrscheinlichkeit von $A, b, c, \bar{c}$ nicht.

ii) Für feste Kozielfunktion ändert die Ersetzung von c durch $-c$, falls die Lösung über dem modifizierten System weiterhin endlich ist, die Wahrscheinlichkeit von $A, b, c, \bar{c}$ nicht.

iii) Die Wahl von $A, b, c, \bar{c}$, so dass es n linear abhängige Spalten in $(c, \bar{c}, A^\top)$ oder $n+1$ linear abhängige Zeilen in (A, b) gibt, hat Wahrscheinlichkeit 0.

Insbesondere sind also fast alle LPs nicht-entartet, d.h. sie haben keine einzige entartete Ecke. Eine uniforme Verteilung hat gewiss diese Eigenschaften, aber nicht nur uniforme Verteilungen.

Man kann die praktische Bedeutung dieser Annahmen in Frage stellen, wenn man realisiert, dass eigentlich alle in der Praxis vorkommenden Probleme mittels einer Modellierungssoftware generiert werden, sehr strukturiert und damit in der Regel hochgradig entartet sind. Dies ist allerdings vor allem die persönliche Sicht des Autors. Doch kommen wir zurück zum Thema:

6.4.2 Proposition. *Sei $x_k x_{k+1}$ eine Kante, die vom Schatteneckenalgorithmus durchlaufen wird. Nach Konstruktion gilt:*

$$c^\top x_{k+1} > c^\top x_k \quad \textit{und} \quad \bar{c}^\top x_{k+1} < \bar{c}^\top x_k, \tag{6.2}$$

Beweis. Weil x_{k+1} optimal bzgl. der Zielfunktion $(\lambda' c + \bar{c})$ ist, gilt

$$(\lambda' c + \bar{c})^\top (x_{k+1} - x_k) > 0. \tag{6.3}$$

Weil x_k optimal bzgl. der Zielfunktion $(\lambda c + \bar{c})$ ist, gilt

$$(-\lambda c - \bar{c})^\top (x_{k+1} - x_k) > 0. \tag{6.4}$$

Addieren wir (6.3) und (6.4), so erhalten wir

$$(\lambda' - \lambda) c^\top (x_{k+1} - x_k) > 0 \tag{6.5}$$

und damit die erste Behauptung.

Multiplizieren wir (6.3) mit $\lambda \geq 0$ und (6.4) mit $\lambda' > 0$ und addieren die so abgeleiteten Ungleichungen, so erhalten wir

$$(\underbrace{\lambda - \lambda'}_{<0}) \bar{c}^\top (x_{k+1} - x_k) > 0 \tag{6.6}$$

und damit den zweiten Teil der Behauptung. □

6.4.3 Proposition. *Findet man eine extremale Richtung y, so ist das Problem unbeschränkt.*

Beweis. Nach Annahme haben wir

$$(\lambda' c + \bar{c})^\top y > 0. \tag{6.7}$$

Weil y keine Extremale in x_k war, schließen wir $(\lambda c + \bar{c})y \leq 0$ und somit

$$(-\lambda c - \bar{c})^\top y \geq 0. \tag{6.8}$$

Addition dieser beiden Ungleichungen ergibt

$$(\underbrace{\lambda' - \lambda}_{>0})c^\top y > 0. \tag{6.9}$$

und damit ist y auch eine extremale Richtung für das Ausgangsproblem. □

Wir nehmen nun einen zufälligen Datensatz $A, b, c, \bar{c}$ her. Nach der Annahme an die Verteilung sind die Spalten von A und b in allgemeiner Lage, ebenso die Zeilen von A, c und $\bar{c}$. Wir betrachten nun die Hyperebenen, die den zulässigen Bereich begrenzen und definieren. Das zugehörige *Hyperebenenarrangement* definiert eine Zerlegung des $\mathbb{K}^n$ in Polyeder. Wir erhalten jedes dieser Polyeder aus den Daten, wenn wir eine Teilmenge der $\leq$-Restriktionen mit -1 multiplizieren. In Abbildung 6.1 haben wir die Situation im Zweidimensionalen skizziert.

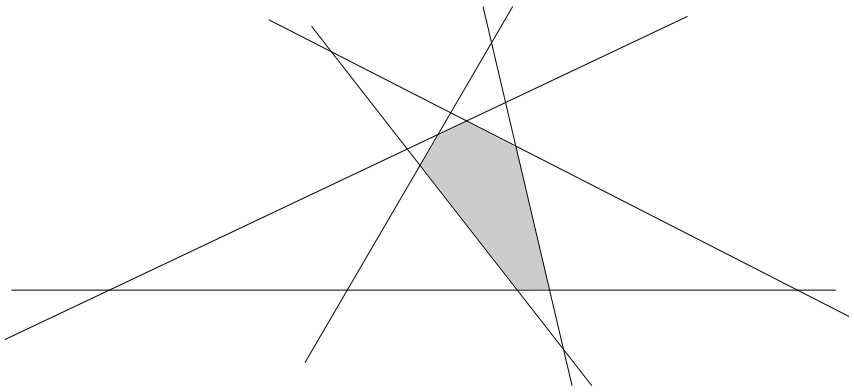


Abbildung 6.1: Ein Linienarrangement zu einem Polytop

Das Linienarrangement dort soll von dem schraffierten Sechseck induziert worden sein. Wir zählen 10 beschränkte Zellen (Polytope) und 12 unbeschränkte Polyeder. Dies ist kein Zufall. Denn in einem affinen Hyperebenenarrangement $\mathscr{H}$

mit $n \geq 1$ Hyperebenen in allgemeiner Lage im $\mathbb{K}^d$ ist die Anzahl $L_b(\mathscr{H})$ der beschränkten volldimensionalen Zellen gegeben durch

$$L_b(\mathscr{H}) = \binom{n-1}{d} \tag{6.10}$$

und die Anzahl $L(\mathscr{H})$ der volldimensionalen Zellen durch

$$L(\mathscr{H}) = \sum_{i=0}^{d} \binom{n}{i}. \tag{6.11}$$

6.4.4 Aufgabe. Beweisen Sie (6.10) und (6.11).

Lösung siehe Lösung 6.7.6.

Wir betrachten im Folgenden nur noch die endlich vielen linearen Optimierungsprobleme, die aus den so durch A, b und Multiplikation der Ungleichungen mit -1 gegebenen Polyedern sowie den Zielfunktionen c und $-c$ definiert sind. Nach Annahme an die Verteilung sind diese Probleme alle gleichwahrscheinlich.

Nun sind wir gewappnet, das folgende Resultat zu beweisen:

6.4.5 Satz (Haimovich 1983)**.** *Die Klasse der LP* $\max_{Ax \leq b} c^\top x$ *kann mit dem Simplexalgorithmus unter Verwendung der Schatteneckenregel unter jeder Wahrscheinlichkeitsverteilung, die die oben gemachten Annahmen erfüllt, mit höchstens* $\frac{n}{2}$ *Iterationen im Durchschnitt gelöst werden.*

Beweis. Wegen der Annahmen an die Wahrscheinlichkeitsverteilung können wir ohne Einschränkung davon ausgehen, dass die Daten in allgemeiner Lage sind. Seien $z_1, \ldots, z_t$ die Basislösungen von $Ax \leq b$, also alle Ecken des Hyperebenenarrangements, da wir auch unzulässige Basislösungen mitzählen. Wegen der allgemeinen Lage ist $t = \binom{m}{n}$.

Sei nun zunächst $\mathscr{L}$ die Klasse der beschränkten LPs, die man aus $\max_{Ax \leq b} c^\top x$ durch Muliplikation einer, eventuell leeren, Teilmenge der Ungleichungen und/oder der Zielfunktion mit -1 erhält. Dann ist $|\mathscr{L}| = 2t$, denn jede Ecke ist Optimallösung für genau jeweils ein LP mit Zielfunktion c und eines mit Zielfunktion $-c$. Diese geometrisch sofort einsichtige Tatsache kann man, wie folgt algebraisch beweisen:

Ist z_i eine solche Ecke, so ist $\operatorname{rg}(A_{\operatorname{eq}(z_i).}) = n$, also ist c Linearkombination der Zeilen von $A_{\operatorname{eq}(z_i).}$. Wegen der allgemeinen Lage der Daten ist die Linearkombination eindeutig und alle Zeilen treten mit nicht verschwindenden Koeffizienten λ_j auf. Mit den Argumenten aus dem Beweis von Proposition 6.4.1 erhalten wir ein

lineares Programm, zu dem z_i mit Zielfunktion c Optimallösung ist, indem wir die Ungleichungen in $\mathrm{eq}(z_i)$ zu den λ_j, die mit negativem Vorzeichen auftreten, mit -1 multiplizieren. Genauso ist z_i Optimallösung des Problems mit Zielfunktion $-c$, wenn wir die Ungleichungen, bei denen die λ_j positiv sind, mit -1 multiplizieren. Weitere Programme in unserer Klasse kann es auf Grund der Eindeutigkeit der Linearkombination und Proposition 6.4.1 nicht geben.

Es sollte Sie nicht irritieren, dass sich die Anzahl der beschränkten linearen Programme nicht mit der Anzahl der beschränkten Zellen aus (6.10) deckt. Das liegt daran, dass ein lineares Programm auch dann beschränkt sein kann, wenn es der zulässige Bereich nicht ist.

Wir wollen nun die Kanten untersuchen. Hier interessiert uns vor allem die Fragestellung, wie oft eine feste Kante von einem der betrachteten linearen Programme vom Simplexalgorithmus unter Verwendung der Schatteneckenregel durchlaufen wird. Die Kanten und Extremalen erhalten wir wegen der allgemeinen Lage, indem wir in $n-1$ der Ungleichungen Gleichheit fordern. Globaler betrachten wir die $\binom{m}{n-1}$ 1-dimensionalen affinen Unterräume des $\mathbb{K}^n$, die man aus dem System (A,b) durch Gleichsetzen von $n-1$ Ungleichungen erhalten kann. Jeder dieser affinen Unterräume enthält $m-n+1$ der z_i, also $m-n$ Kanten und zwei Extremalen.

Sind z_i und z_j benachbarte Ecken auf einer solchen affinen Geraden, so behaupten wir:

Die Strecke $\overline{z_iz_j}$ wird von höchstens einem LP aus $\mathscr{L}$ durchlaufen.

Ist nämlich $\max_{\tilde{A}x\leq\tilde{b}}\tilde{c}^\top x$ ein solches Programm, so gibt es wegen Proposition 6.4.2 ein $\lambda>0$, so dass

$$\overline{z_iz_j}=\{x\in\mathbb{R}^n\mid \tilde{A}x\leq\tilde{b},(\bar{c}+\lambda\tilde{c})^\top x=(\bar{c}+\lambda\tilde{c})^\top z_i\}$$

und es gilt $(\bar{c}+\lambda\tilde{c})^\top x\leq(\bar{c}+\lambda\tilde{c})^\top z_i$ für alle $x\in P(\tilde{A},\tilde{b})$. Wegen der allgemeinen Lage sind alle bis auf $n-1$ Ungleichungen im Inneren von $\overline{z_iz_j}$ strikt, also festgelegt. Da das Programm auf der Strecke $\overline{z_iz_j}$ optimal, also konstant, ist, und $\mathrm{rg}(A_{\mathrm{eq}(\overline{z_iz_j})\cdot})=n-1$ ist, muss $\bar{c}+\lambda c$ Linearkombination der Zeilen von $A_{\mathrm{eq}(\overline{z_iz_j})\cdot}$ sein. Also liefern uns, analog zum Fall der Ecken, die Koeffizienten der Darstellung von $\bar{c}+\lambda\tilde{c}$ als Linearkombination der Zeilen von $A_{\mathrm{eq}(\overline{z_iz_j})\cdot}$ mittels Proposition 6.4.1 die Richtung der Ungleichung auf $\mathrm{eq}(\overline{z_iz_j})$. Diese Kante wird folglich in höchstens einem Programm aus $\mathscr{L}$ durchlaufen. Beachte, $\bar{c}+\lambda c$ ist zwar hier nicht mehr in allgemeiner Lage, aber wegen der allgemeinen Lage von c und $\bar{c}$ darf keiner der

$n-1$ Koeffizienten in der Linearkombination, die diesen Vektor in den Zeilen der Kantengleichungen darstellt, verschwinden.

Analog wird jede Extremale von höchstens einem Programm in $\mathscr{L}$ betreten. Von den zwei Extremalen einer dieser affinen Geraden entfällt eine wegen $\bar{c}^\top y > 0$, denn wie im Beweis von Proposition 6.4.3 schließen wir zunächst $(\lambda' - \lambda)\bar{c}^\top y < 0$ und somit gilt wegen $\lambda' > \lambda$ für einen gefundenen Strahl $\bar{c}^\top y < 0$.

Fassen wir die Resultate zusammen. Wir betrachten alle linearen Programme die durch die Daten A, b, c definiert werden. Das sind mehr als $|\mathscr{L}| = 2\binom{m}{n}$, die Anzahl der beschränkten linearen Programme. Jede der $(m-n)\binom{m}{n-1}$ Kanten des Hyperebenenarrangements wird für höchstens ein lineares Programm bei Verwendung des Schatteneckenalgorithmus durchlaufen. Das gleiche gilt für $\binom{m}{n-1}$ Extremalen. Also ist die durchschnittliche Anzahl an Iterationen eines LP in $\mathscr{L}$ nach oben beschränkt durch

$$\begin{aligned}\frac{(m-n+1)\binom{m}{n-1}}{|\mathscr{L}|} &= \frac{\frac{(m-n+1)m!}{(m-n+1)!(n-1)!}}{2\frac{m!}{n!(m-n)!}} \\ &= \frac{m!n!(m-n)!}{2(m-n)!(n-1)!m!} \\ &= \frac{n}{2}.\end{aligned}$$

□

Wie versprochen geben wir nun noch Resultate für weitere Klassen von LPs ohne Beweis an.

6.4.6 Satz (Haimovich 1983)**.** *Sei $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$. Dann gelten folgende obere Schranken für Erwartungswerte an die Anzahl der Iterationen für das Simplexverfahren mit der Schatteneckenregel, jeweils mit den obigen Verteilungsannahmen*

i) $\left(\frac{2}{n} + \frac{2}{m+1}\right)^{-1}$ *für die Problemklasse* $\max\{c^\top x \mid Ax \le b, x \ge 0\}$,

ii) $\left(\frac{2}{n} + \frac{2}{m-n+1}\right)^{-1}$ *für die Problemklasse* $\max\{c^\top x \mid Ax \le b\}$,

iii) $\left(\frac{2}{n-m} + \frac{2}{m+1}\right)^{-1}$ *für die Problemklasse* $\max\{c^\top x \mid Ax = b, x \ge 0\}$.

Beweis. siehe Mordecai Haimovich, "The Simplex Algorithm Is Very Good!: On the Expected Number of Pivot Steps and Related Properties of Random Linear Programs" (February 1996). Discussion Paper 99, Center for Rationality and Interactive Decision Theory, The Hebrew University of Jerusalem. □

6.4.7 Bemerkung. Goldfarb [9] gab 1983 ein Beispiel an, dass Borgwardts Schatteneckenalgorithmus auch keine polynomiale Pivotregel liefert.

6.4.8 Beispiel. Wir lösen das folgende lineare Programm mit dem Schatteneckenalgorithmus. Maximiere

$$-2\xi_1 + \xi_2 + \xi_3$$

unter der Bedingung

$$\begin{array}{rcrcrcl} \xi_1 & - & \xi_2 & + & \xi_3 & \leq & 2 \\ -\xi_1 & + & \xi_2 & + & \xi_3 & \leq & 2 \\ & & & & \xi_1, \xi_2, \xi_3 & \geq & 0. \end{array}$$

Als Startecke sei $(0,0,0)^\top$ gegeben und als Kozielfunktion

$$\min \xi_1 + \xi_2 + \xi_3.$$

Das ergibt dann das Startableau

$$\begin{array}{rrrrr|l} -2 & 1 & 1 & 0 & 0 & 0 \\ \hline -1 & -1 & -1 & 0 & 0 & 0 \\ \hline 1 & -1 & 1 & 1 & 0 & 2 \\ -1 & \boxed{1} & 1 & 0 & 1 & 2. \end{array}$$

Wir addieren nun vorsichtig ein Vielfaches der eigentlichen Zielfunktion zur Kozielfunktion, bis ein Eintrag der reduzierten Kosten Null wird.

Bei der Anwendung der Schatteneckenregel tritt Entartung in der Wahl des neuen Basiselementes auf, die Daten sind auch offensichtlich nicht in allgemeiner Lage. Wir entscheiden uns für die zweite Spalte. Das nächste Tableau ist dann, da $\lambda = 1$ ist

$$\begin{array}{rrrrr|l} -1 & 0 & 0 & 0 & -1 & -2 \\ \hline -3 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 2 & 1 & 1 & 4 \\ -1 & 1 & 1 & 0 & 1 & 2 \end{array}$$

und die Ecke ist nun auch optimal bzgl. der Ausgangszielfunktion. Also ist $(0,2,0)^\top$ eine Optimallösung und der Optimalwert 2.

6.4.9 Bemerkung. Zu Beginn dieses Jahrtausends haben Spielmann und Teng [22] eine weitere Analysevariante der Komplexität des Simplexalgorithmus vorgestellt, die ein Mittelding zwischen worst-case-Komplexität und mittlerer Laufzeit bilden soll. Dabei werden die linearen Programme und lokal perturbiert und die Laufzeit über die Umgebungen gemittelt und dann das Supremum aller dieser Werte gebildet. Dieses Modell teilt mit dem oben ausgeführten das Problem, dass entartete Programme im Wesentlichen nicht berücksichtigt werden.

6.5 Dantzig-Wolfe Dekomposition

Zum Abschluss dieses Kapitels wollen wir noch eine Dekompositionstechnik vorstellen. Oftmals hat man es mit linearen Optimierungsproblemen zu tun, bei denen ein Teilproblem gutartig ist. Eventuell kennt man für dieses sogar einen streng polynomialen Algorithmus.

Wir wollen hier Lineare Programme betrachten, bei denen ein Teil der Restriktionsmatrix eine einfache Struktur hat, also von dem Typ

$$\begin{array}{lrcl} \max & c^\top x & & \\ \text{unter} & \binom{A}{D} x & = & \binom{b}{d} \\ & x & \geq & 0. \end{array}$$

Zusätzlich nehmen wir an, dass das Polyeder $P = \{x \in \mathbb{R}^n_+ \mid Dx = d\}$ beschränkt ist, und die Menge seiner Ecken bekannt ist oder zumindest leicht zu bestimmen ist, etwa weil das Teilproblem streng polynomial lösbar ist. Wir bezeichnen die Ecken dieses Teilproblems mit $\{v_1, \ldots, v_t\}$. Dann können wir das Programm umschreiben zu:

$$\begin{array}{lll} & \max & \sum_{i=1}^t \lambda_i c^\top v_i \\ \text{(T)} & \text{unter} & \sum_{i=1}^t \lambda_i A v_i = b \\ & & \sum_{i=1}^t \lambda_i = 1 \\ & & \lambda \geq 0. \end{array}$$

Da üblicherweise die Anzahl der Ecken des Unterproblems zu groß ist, als dass man dieses Programm explizit lösen könnte, werden wir versuchen, die „richtigen“ Ecken dafür auszuwählen. Beachte, dass in obigem Problem die Spalten der Restriktionsmatrix das Aussehen $\binom{Av_i}{1}$ haben und die λ_i die Variablen sind. Eine Basis ist also gegeben durch $m+1$ Ecken v_i. Schreiben wir die $((m+1) \times t)$-Matrix, deren Spalten die $\binom{Av_i}{1}$ sind als $\hat{A}$, und den analogen Kostenvektor als $\hat{c} \in \mathbb{R}^t$, nennen die

Basis B und betrachten wir unser Optimalitätskriterium, so lautet dieses

$$\hat{c}^\top - \hat{c}_B^\top \hat{A}_{.B}^{-1} \hat{A} \leq 0.$$

Den Teil $\hat{c}_B^\top \hat{A}_{.B}^{-1}$ der reduzierten Kosten kennen wir auch als Dualvariablen zu den Restriktionen des Programms, wir können sie mit der Basis als bekannt annehmen. Eine dieser Dualvariablen gehört zu der Restriktion $\sum_{i=1}^{t} \lambda_i = 1$, nennen wir diese α und die übrigen w, so können wir $(w^\top, \alpha) = \hat{c}_B^\top \hat{A}_{.B}^{-1}$ als bekannt annehmen. Das Optimalitätskriterium können wir dann überprüfen, indem wir

$$\max_{i=1}^{t} c^\top v_i - (w^\top, \alpha) \begin{pmatrix} Av_i \\ 1 \end{pmatrix}$$

bestimmen, wobei α eine, von Subproblem zu Subproblem variierende Konstante ist. Hierfür haben wir also das lineare Teilproblem

$$\begin{array}{lll} & \max & c^\top x - w^\top Ax \\ \text{unter} & Dx & = d \\ & x & \geq 0 \end{array}$$

zu lösen. Dies ist auf Grund der angenommenen einfachen Struktur des Teilproblems leicht und wir können durch einen Schritt des revidierten Simplexverfahrens eine Spalte mit positiven reduzierten Kosten neu in die Basis aufnehmen. Da das Subproblem und somit auch das gesamte Problem beschränkt ist, können wir ein basisverlassendes Element bestimmen und iterieren so lange, bis die Lösung des Teilproblems beweist, dass die aktuelle Basis optimal für das Gesamtproblem ist. Wir fassen zusammen:

Schematische Skizze des Dekompositionsverfahrens nach Dantzig-Wolfe:

Eingabedaten: $A \in \mathbb{R}^{m \times n}$ mit vollem Zeilenrang, $Y \in \mathbb{R}^{n \times k}$, ein Polytop $X = \text{Conv}(Y)$, $b \in \mathbb{R}^m, b \geq 0$, $\hat{A}_{.B} = \binom{AY}{1}$ als zulässige Basis, $c \in \mathbb{R}^n$.

Berechne die Dualvariablen $(w^\top, \alpha)$ und $v_k \in \text{argmax}_{x \in X} c^\top x - w^\top Ax$.

Solange $c^\top v_k - w^\top Av_k > \alpha$**:**

Zeilenwahl: Bestimme basisverlassendes Element v_j als Argument von

$$\min \left\{ \frac{\left(\hat{A}_{.B}^{-1} \binom{b}{1}\right)_r}{\left(\hat{A}_{.B}^{-1} \binom{Av_k}{1}\right)_r} \mid \left(\hat{A}_{.B}^{-1} \begin{pmatrix} Av_k \\ 1 \end{pmatrix}\right)_r > 0 \right\}.$$

Basiswechsel: Nehme v_k in die Basis auf und entferne v_j.

Spaltenwahl: Berechne die Dualvariablen $(w^\top, \alpha)$ und

$$v_k \in \arg\max_{x \in X} c^\top x - w^\top A x.$$

6.5.1 Bemerkung. i) Unter Berücksichtigung der Transformation des Teilproblems handelt es sich um eine direkte Implementierung des revidierten Simplexalgorithmus für das Problem (T), bei der die Berechnung der reduzierten Kosten ausgelagert wurde. Unter der Annahme, dass kein Zykeln auftritt (oder durch Maßnahmen wie Zykelerkennung und Perturbation oder andere Methoden verhindert wird), folgt sofort Endlichkeit und Korrektheit des Algorithmus.

ii) Man nennt diese Methode auch "Column Generation", da in jedem Schritt die Spalte der Matrix für die neue Basisvariable erst erzeugt werden muss.

iii) Bei der Lösung des Teilproblems ändern sich jeweils nur die Dualvariablen des Masterproblems und damit die Kostenfunktion. Dies ermöglicht bei manchen Teilproblemen einen „Warmstart".

iv) Es ist nicht unbedingt nötig, in jedem Schritt das Teilproblem bis zur Optimalität zu lösen, da es genügt, eine Variable mit positiven reduzierten Kosten zu bestimmen.

Vor einem numerischen Beispiel wollen wir noch eine obere Schranke für die Berechnungen bestimmen. Auf Grund der Vielzahl an Variablen kann es sein, dass die Bestimmung einer Optimallösung des Problems zu aufwändig erscheint und man stattdessen mit einer gewissen garantierten Qualität der Lösung zufrieden ist. Da das Verfahren stets zulässige Lösungen produziert, die also somit eine untere Schranke für die Optimallösung sind, wäre das also der Fall, wenn der Unterschied zwischen unterer und oberer Schranke (absolut oder relativ) einen gewissen Schwellwert unterschreitet.

6.5.2 Proposition. *Sei w die Menge der Dualvariablen zu einer Basis B des Problems und k Index einer Optimallösung des zugehörigen Teilproblems. Dann ist*

$$\max_{\substack{Ax=b \\ x \in X}} c^\top x \leq \hat{c}_B \hat{A}_{.B}^{-1} \begin{pmatrix} b \\ 1 \end{pmatrix} + c^\top v_k - w^\top A v_k - \alpha.$$

Beweis.

$$\begin{aligned} c^\top v_k - w^\top A v_k - \alpha &\overset{v_k \text{ opt.}}{\geq} c^\top x - w^\top A x - \alpha \\ \iff c^\top x &\leq w^\top A x - w^\top A v_k + c^\top v_k \\ \iff c^\top x &\leq w^\top b - w^\top A v_k + c^\top v_k \\ \iff c^\top x &\overset{(w,\alpha)=c_B \hat{A}_{.B}^{-1}}{\leq} \hat{c}_B \hat{A}_{.B}^{-1} \begin{pmatrix} b \\ 1 \end{pmatrix} - \alpha - w^\top A v_k + c^\top v_k. \end{aligned}$$

□

Da $\hat{c}_B \hat{A}_{.B}^{-1} \binom{b}{1}$ der Zielfunktionswert der aktuellen Lösung ist und $c^\top v_k - w^\top A v_k$ bis auf α der Zielfunktionswert des Teilproblems ist, kann man den Unterschied zwischen oberer und unterer Schranke leicht berechnen, bzw. kommt dieser in den Berechnungen sowieso vor.

6.5.3 Beispiel. Wir betrachten das Problem

$$\begin{array}{llrcrcrcrcl} \max & 2x_1 & + & x_2 & + & x_3 & - & x_4 & & \\ \text{unter} & x_1 & & & + & x_3 & & & \leq & 2 \\ & x_1 & + & x_2 & & & + & 2x_4 & \leq & 3 \\ & x_1 & & & & & & & \leq & 2 \\ & x_1 & + & 2x_2 & & & & & \leq & 5 \\ & & & & - & x_3 & + & x_4 & \leq & 2 \\ & & & & & 2x_3 & + & x_4 & \leq & 6 \\ & & & & & & & x & \geq & 0 \end{array}$$

Die letzten vier Restriktionen haben eine einfache Struktur, da die dritte und vierte Restriktion nur x_1 und x_2 und die fünfte und sechste Restriktion nur x_3 und x_4 betreffen. Wir setzen also

$$X := \left\{ x \in \mathbb{R}^4 \;\middle|\; \begin{array}{rcrcrcrcl} x_1 & & & & & & & \leq & 2 \\ x_1 & + & 2x_2 & & & & & \leq & 5 \\ & & & - & x_3 & + & x_4 & \leq & 2 \\ & & & & 2x_3 & + & x_4 & \leq & 6 \\ & & & & & & x & \geq & 0. \end{array} \right\}$$

Dann ist X das Kartesische Produkt der beiden zweidimensionalen Polygone in Abbildung 6.2. Also ist X ein vierdimensionales Polytop mit 16 Ecken $v_1, \ldots, v_{16}$.

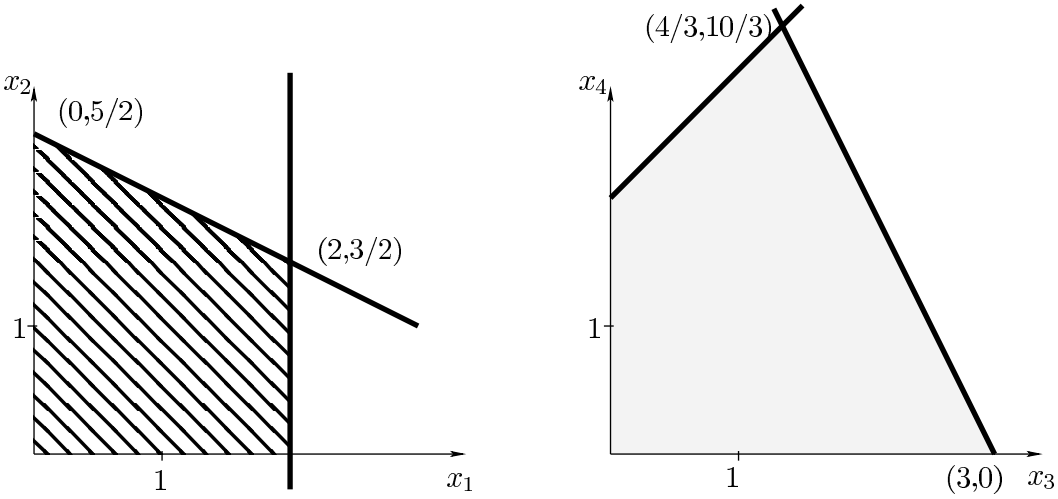


Abbildung 6.2: Die Menge X im Beispiel

Unser Masterproblem lautet also:

$$\begin{array}{rrcl} & \max \sum_{i=1}^{16} \hat{c}_i \lambda_i & & \\ \text{unter} & \sum_{i=1}^{16} (Av_i)\lambda_i + y & = & b \\ & \sum_{i=1}^{16} \lambda_i & = & 1 \\ & \lambda, y & \geq & 0 \end{array}$$

Als Startecke wählen wir den Punkt $v_1 = (0,0,0,0)^\top$, also die Schlupfvariablen in der Basis. Dann ist $(w^\top, \alpha) = (0^\top, 0)$ und $\bar{b} = \binom{b}{1}$. Unser Tableau lautet jetzt, da nur die Ecke v_1 beteiligt ist:

z	0	0	0	0
y_1	1	0	0	2
y_2	0	1	0	3
λ_1	0	0	1	1.

Als Teilproblem müssen wir jetzt also $\max_{x \in X} c^\top x$ lösen. Dieses Problem können wir mittels Hinsehen lösen, da es in zwei unabhängige zweidimensionale Probleme separiert. Die Optimallösung ist $v_2 = (2, 3/2, 3, 0)^\top$ vom Wert $\frac{17}{2} > 0$. Also generieren wir die zugehörige Spalte und nehmen sie in die Basis auf:

$$Av_2 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 2 \end{pmatrix} \begin{pmatrix} 2 \\ \frac{3}{2} \\ 3 \\ 0 \end{pmatrix} = \begin{pmatrix} 5 \\ \frac{7}{2} \end{pmatrix}.$$

Somit ist, da $\hat{A}_{.B}$ die Einheitsmatrix ist, unsere neue Spalte $(\frac{17}{2},5,\frac{7}{2},1)^\top$.

z	0	0	0	0	$\frac{17}{2}$
y_1	1	0	0	2	$\boxed{5}$
y_2	0	1	0	3	$\frac{7}{2}$
λ_1	0	0	1	1	1

Wir lesen als obere Schranke $\frac{17}{2}-0=8.5$ ab. Der Minimalquotiententest identifiziert 5 als Pivotelement und wir erhalten als neues Tableau:

z	$-\frac{17}{10}$	0	0	$-\frac{17}{5}$
λ_2	$\frac{1}{5}$	0	0	$\frac{2}{5}$
y_2	$-\frac{7}{10}$	1	0	$\frac{8}{5}$
λ_1	$-\frac{1}{5}$	0	1	$\frac{3}{5}$

Dies gehört zu der Lösung $\frac{3}{5}(0,0,0,0)^\top+\frac{2}{5}(2,\frac{3}{2},3,0)^\top=\frac{1}{5}(4,3,6,0)^\top$ mit Zielfunktionswert $\frac{17}{5}=3.4$. Wir haben $(w_1,w_2,\alpha)=(\frac{17}{10},0,0)$.

Also berechnen wir als Zielfunktion für das Teilproblem

$$(2,1,1,-1)-(\frac{17}{10},0)\begin{pmatrix}1&0&1&0\\1&1&0&2\end{pmatrix}=(\frac{3}{10},1,-\frac{7}{10},-1).$$

Als Lösung unseres Teilproblems lesen wir aus Abbildung 6.2 $v_3=(0,\frac{5}{2},0,0)^\top$ mit Zielfunktionswert $\frac{5}{2}$ ab und berechnen

$$Av_3=\begin{pmatrix}0\\\frac{5}{2}\end{pmatrix}.$$

Wir berechnen

$$\hat{A}_{.B}^{-1}\begin{pmatrix}Av_3\\1\end{pmatrix}=\begin{pmatrix}\frac{1}{5}&0&0\\-\frac{7}{10}&1&0\\-\frac{1}{5}&0&1\end{pmatrix}\begin{pmatrix}0\\\frac{5}{2}\\1\end{pmatrix}=\begin{pmatrix}0\\\frac{5}{2}\\1\end{pmatrix}.$$

Somit ist unsere neue Spalte $(\frac{5}{2},0,\frac{5}{2},1)^\top$.

z	$-\frac{17}{10}$	0	0	$-\frac{17}{5}$	$\frac{5}{2}$
λ_2	$\frac{1}{5}$	0	0	$\frac{2}{5}$	0
y_2	$-\frac{7}{10}$	1	0	$\frac{8}{5}$	$\frac{5}{2}$
λ_1	$-\frac{1}{5}$	0	1	$\frac{3}{5}$	$\boxed{1}$

Wir lesen als obere Schranke ab $\frac{5}{2}+\frac{17}{5}=5.9$ ab und nach dem Pivot erhalten wir:

z	$-\frac{6}{5}$	0	$-\frac{5}{2}$	$-\frac{49}{10}$
λ_2	$\frac{1}{5}$	0	0	$\frac{2}{5}$
y_2	$-\frac{1}{5}$	1	$-\frac{5}{2}$	$\frac{1}{10}$
λ_3	$-\frac{1}{5}$	0	1	$\frac{3}{5}$.

Hierzu gehört die Lösung

$$\frac{2}{5}(2,\frac{3}{2},3,0)^\top+\frac{3}{5}(0,\frac{5}{2},0,0)^\top=(\frac{4}{5},\frac{21}{10},\frac{6}{5},0)^\top$$

mit Zielfunktionswert $\frac{49}{10}=4.9$. Wir berechnen die Zielfunktion des Teilproblems

$$(2,1,1,-1)-(\frac{6}{5},0)\begin{pmatrix}1&0&1&0\\1&1&0&2\end{pmatrix}=(\frac{4}{5},1,-\frac{1}{5},-1)$$

und lesen als Optimallösung $v_4=(2,\frac{3}{2},0,0)^\top$ ab. Da $\alpha=\frac{5}{2}$ ist, haben wir immer noch positive reduzierte Kosten, nämlich $\frac{8}{5}+\frac{3}{2}-\frac{5}{2}=\frac{3}{5}$. Wir berechnen

$$Av_4=\begin{pmatrix}1&0&1&0\\1&1&0&2\end{pmatrix}\begin{pmatrix}2\\\frac{3}{2}\\0\\0\end{pmatrix}=\begin{pmatrix}2\\\frac{7}{2}\end{pmatrix}$$

und

$$\hat{A}_{.B}^{-1}\begin{pmatrix}Av_4\\1\end{pmatrix}=\begin{pmatrix}\frac{1}{5}&0&0\\-\frac{1}{5}&1&-\frac{5}{2}\\-\frac{1}{5}&0&1\end{pmatrix}\begin{pmatrix}2\\\frac{7}{2}\\1\end{pmatrix}=\begin{pmatrix}\frac{2}{5}\\\frac{3}{5}\\\frac{3}{5}\end{pmatrix}.$$

Somit ist unsere neue Spalte $(\frac{3}{5},\frac{2}{5},\frac{3}{5},\frac{3}{5})^\top$.

z	$-\frac{6}{5}$	0	$-\frac{5}{2}$	$-\frac{49}{10}$	$\frac{3}{5}$
λ_2	$\frac{1}{5}$	0	0	$\frac{2}{5}$	$\frac{2}{5}$
y_2	$-\frac{1}{5}$	1	$-\frac{5}{2}$	$\frac{1}{10}$	$\boxed{\frac{3}{5}}$
λ_3	$-\frac{1}{5}$	0	1	$\frac{3}{5}$	$\frac{3}{5}$.

Wir lesen als obere Schranke ab $\frac{3}{5}+\frac{49}{10}=\frac{11}{2}=5.5$. Man beachte, dass die reduzierten Kosten der generierten Spalte stets die Differenz zwischen aktuellem besten Zielfunktionswert und oberer Schranke darstellen. Nach dem Pivot erhalten wir.

z	-1	-1	0	-5
λ_2	$\frac{1}{3}$	$-\frac{2}{3}$	$\frac{5}{3}$	$\frac{1}{3}$
λ_4	$-\frac{1}{3}$	$\frac{5}{3}$	$-\frac{25}{6}$	$\frac{1}{6}$
λ_3	0	-1	$\frac{7}{2}$	$\frac{1}{2}$

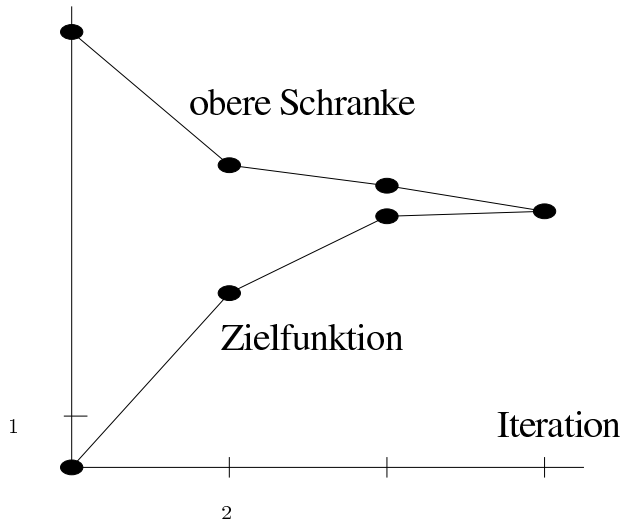


Abbildung 6.3: Obere und untere Schranke

Hierzu gehört die Lösung

$$\frac{1}{3}(2,\frac{3}{2},3,0)^\top+\frac{1}{2}(0,\frac{5}{2},0,0)^\top+\frac{1}{6}(2,\frac{3}{2},0,0)^\top=(1,2,1,0)^\top$$

mit Zielfunktionswert 5 und wir lesen ab $(w^\top,\alpha)=(1,1,0)$. Wir berechnen die Zielfunktion des Teilproblems

$$(2,1,1,-1)-(1,1)\begin{pmatrix}1&0&1&0\\1&1&0&2\end{pmatrix}=(0,0,0,-3)$$

und lesen als Lösung des Teilproblems $v_5=(0,0,0,0)^\top$ ab. Da $\alpha=0$ ist, haben wir Optimalität in der Ecke $(1,2,1,0)^\top$ erreicht. In Abbildung 6.3 haben wir die Entwicklung von Zielfunktion und oberer Schranke aus Proposition 6.5.2 geplottet.

6.5.4 Bemerkung. i) Eine zulässige Startlösung kann man, ähnlich wie beim gewöhnlichen Simplex, mittels Zweiphasen- oder Big-M-Methode bestimmen.

ii) Im Falle eines unbeschränkten Bereiches X muss man das Verfahren so modifizieren, dass Punkte in $X=\mathrm{Conv}(V)+\mathrm{Cone}(E)$ generiert werden.

6.6 Anhang: Die Landau-Symbole

Bei der Abschätzung von Laufzeiten von Algorithmen oder bei der Messung von Konvergenzgeschwindigkeiten benutzt man üblicherweise die Landau-Symbole.

Damit kann man auf bequeme Weise Aussagen über das asymptotische Verhalten reeller Folgen machen, wobei wir z.B. ganzzahlige Folgen als Spezialfälle reeller Folgen betrachten.

Wir werden hier etwas ausführlicher auf die Abschätzung nach oben, die als Big-Oh-Notation bekannt ist, eingehen und die anderen Symbole am Schluß nur definieren.

6.6.1 Definition. Seien $f, g : \mathbb{N} \to \mathbb{R}$ Abbildungen. Dann schreiben wir

$$f = O(g)$$

oder

$$f(n) = O(g(n)),$$

wenn es eine Konstante C und einen Startpunkt $n_1 \in \mathbb{N}$ gibt, so dass für alle $n \in \mathbb{N}$, $n \geq n_1$, gilt $|f(n)| \leq Cg(n)$.

Vorsicht! Die „Big-Oh"-Notation liefert nur eine Abschätzung nach oben, nicht nach unten. Zum Beispiel ist $n = O(n^5)$. Insbesondere ist das hier benutzte Gleichheitszeichen nicht kommutativ. Diese Schreibweise ist aber allgemein üblich und deswegen verwenden wir sie ebenso.

Folgende Zusammenhänge sind nützlich bei Abschätzungen (z. B. auch von Laufzeiten von Algorithmen).

6.6.2 Proposition. *Seien $C, a, \alpha, \beta > 0$ feste reelle, positive Zahlen unabhängig von n. Dann gilt*

i) $\alpha \leq \beta \Rightarrow n^\alpha = O(n^\beta)$,

ii) $a > 1 \Rightarrow n^C = O(a^n)$,

iii) $\alpha > 0 \Rightarrow (\ln n)^C = O(n^\alpha)$.

Beweis.

i) Wir haben zu zeigen, dass $n^\alpha \leq Cn^\beta$ zumindest ab einem gewissen n_0 gilt. Da aber $n^\beta = n^\alpha \underbrace{n^{\beta-\alpha}}_{\geq 1}$ haben wir sogar stets $n^\alpha \leq n^\beta$ mit der Konstanten $C = 1$.

ii) Wir betrachten die Folge

$$a_n := \left(\frac{n}{n-1}\right)^C.$$

Nach den Grenzwertsätzen und wegen der Stetigkeit der Exponentialfunktion ist $\lim_{n\to\infty} a_n = 1$. Da $a > 1$ ist, gibt es für $\varepsilon = a-1$ ein $n_1 \in \mathbb{N}$, so dass für alle $n \geq n_1$ gilt $|a_n - 1| < \varepsilon = a-1$, also insbesondere

$$\forall n \geq n_1 : a_n = a_n - 1 + 1 \leq |a_n - 1| + 1 < a - 1 + 1 = a.$$

Nun setzen wir

$$C_1 := \frac{n_1^C}{a^{n_1}}$$

und zeigen

$$n^C \leq C_1 a^n \tag{6.12}$$

mittels vollständiger Induktion für $n \geq n_1$. Zu Anfang haben wir

$$n_1^C = C_1 a^{n_1}.$$

Sei also $n > n_1$. Dann ist unter Ausnutzung der Induktionsvoraussetzung und wegen $a_n \leq a$

$$n^C = \left(\frac{n}{n-1}\right)^C (n-1)^C = a_n (n-1)^C \overset{IV}{\leq} a_n C_1 a^{n-1} \overset{a_n \leq a}{\leq} a C_1 a^{n-1} = C_1 a^n.$$

Also gilt (6.12) für $n \geq n_1$, also per definitionem $n^C = O(a^n)$.

iii) Wir setzen $a := e^\alpha$. Dann ist $a > 1$ und wir wählen n_1 und C_1 wie eben. Ferner wählen wir n_2 mit $\ln(n_2) \geq n_1$. Indem wir die Monotonie und Stetigkeit des Logarithmus ausnutzen, erhalten wir für $n \geq n_2$ nach b)

$$\begin{aligned} & (\ln n)^C \leq C_1 a^{\ln n} \\ \iff & (\ln n)^C \leq C_1 (e^{\ln a})^{\ln n} = C_1 (e^{\ln n})^{\ln a} = C_1 n^{\ln a} \\ \iff & (\ln n)^C \leq C_1 n^\alpha. \end{aligned}$$

□

Wir merken uns, dass Logarithmen langsamer wachsen als Wurzel- und Polynomfunktionen und diese wiederum langsamer als Exponentialfunktionen.

6.6.3 Beispiel. Wenn man eine Formelsammlung zur Hand hat, schlägt man nach (und beweist mittels vollständiger Induktion), dass

$$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}. \tag{6.13}$$

Hat man keine Formelsammlung zur Hand, ist die Herleitung dieser Formel recht mühselig. Darum schätzen wir ab: Zunächst ist $\sum_{i=1}^{n} i^3 \leq \sum_{i=1}^{n} n^3 = n^4$. Außerdem ist $\sum_{i=1}^{n} i^3 \geq \sum_{i=\lfloor \frac{n}{2} \rfloor}^{n} \left(\frac{n}{2}\right)^3 \geq \frac{n^4}{16}$. Also verhält sich die Summe „bis auf einen konstanten Faktor“ wie n^4.

Im Falle des Beispiels ist n^4 nicht nur eine obere, sondern auch eine untere Schranke. Auch dafür gibt es Symbole wie z. B.

$f(n) = o(g(n)) \;\; :\Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, also wächst f echt langsamer als g,

$f(n) = \Omega(g(n)) \;\; :\Leftrightarrow g(n) = O(f(n))$, $g(n)$ ist eine untere Schranke für $f(n)$ für große n,

$f(n) = \Theta(g(n)) \;\; :\Leftrightarrow f(n) = O(g(n))$ und $f(n) = \Omega(g(n))$, also verhalten sich f und g „bis auf einen konstanten Faktor“ asymptotisch gleich, genauer gibt es $c_1, c_2 > 0$ und $n_0 \in \mathbb{N}$ mit

$$\forall n \geq n_0 : c_1 g(n) \leq f(n) \leq c_2 g(n).$$

$f(n) \sim g(n) \;\; :\Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$, wie eben, aber „exakt“ mit Faktor 1.

6.7 Lösungsvorschläge zu den Übungen

6.7.1 Lösung (zu Aufgabe 6.1.2)**.**
Eingangsgrößen sind für uns $A \in \mathbb{Z}^{m\times n}, b \in \mathbb{Z}^m, c \in \mathbb{Z}^n$. Das sind $mn+m+n$ Zahlen. Also ist

$$I(w) = mn+m+n = O(nm).$$

Für die andere Maßgröße setzen wir zunächst

$$C = \max\left\{|A_{ij}|, |c_j|, |b_i| \mid 1 \le i \le m, 1 \le j \le n\right\} + 1.$$

Dann ist

$$\begin{aligned}\langle w\rangle &= \sum_{i=1}^{m}\sum_{j=1}^{n}(1+\lceil\log_2(|A_{ij}|+1)\rceil) + \sum_{i=1}^{m}(1+\lceil\log_2(|b_i|+1)) \\ &\quad + \sum_{j=1}^{n}(1+\lceil\log_2(|c_j|+1)\rceil) \\ &= O(\log_2(C+1)mn).\end{aligned}$$

6.7.2 Lösung (zu Aufgabe 6.1.3)**.**
Sei C der Betrag der größten in den Berechnungen auf w vorkommenden Zahl plus 1. Dann ist

$$\langle w\rangle = O(\log_2(C)I(w)).$$

Sei n^k ein Polynom, das asymptotisch eine obere Schranke für die Anzahl der elementaren Rechenoperationen des streng polynomialen Algorithmus liefert, also mit

$$t_M^s(n) = O(n^k).$$

Sei n^l ein Polynom, das asymptotisch eine obere Schranke für die Rechenzeit der elementaren Rechenoperationen liefert. Dann ist

$$\begin{aligned}t_M(n) &= t_M(< w >) \\ &\le t_M((1+\lceil\log_2(C)\rceil)I(w)) \\ &= O((1+\lceil\log_2(C)\rceil)^{kl}I(w)^k) \\ &= O(n^{k+kl}).\end{aligned}$$

ein Polynom, das die Laufzeit des Algorithmus begrenzt.

Man beachte, dass in dieser Schreibweise der O-Notation die Gleichheit nicht kommutativ ist. So ist etwa $O(n^k) = O(n^{k+1})$, aber $O(n^{k+1}) \neq O(n^k)$. Manche Autoren ersetzen daher das Gleichheitssymbol durch ein Element oder Inklusionssymbol, da es sich auf der rechten Seite streng genommen um eine Funktionenklasse handelt. Wir verwenden hier die üblichere, aber unsauberere Schreibweise.

6.7.3 Lösung (zu Aufgabe 6.2.7)**.**
Wir zeigen zunächst, dass der euklidische Algorithmus zur Bestimmung des größten gemeinsamen Teilers zweier Zahlen $a_1, a_2 \in \mathbb{N}$ nach höchstens

$$2\lceil \max\{\log_2(a_1), \log_2(a_2)\}\rceil + 1$$

Iterationen terminiert.

Seien dazu $a_1, a_2 \in \mathbb{N}$ gegeben, auch alle weiteren a_i, q_i seien stets positive natürliche Zahlen. O.E. sei $a_1 > a_2$, andernfalls führt die erste Iteration des euklidischen Algorithmus eine Vertauschung der Elemente durch oder der Algorithmus terminiert, da $a_1 = a_2$ ist.

Seien im Folgenden dann für $1 \leq i \leq k-1$ die Zahlen $0 < a_{i+2} < a_{i+1}$ definiert durch $a_i = q_i a_{i+1} + a_{i+2}$ und $a_k = q_k a_{k+1}$. Dann ist bekanntlich a_{k+1} der größte gemeinsame Teiler von a_1 und a_2. Wir behaupten nun

$$\forall i = 1, \ldots, k-1 : a_{i+2} < \frac{1}{2} a_i.$$

Wenn wir zusätzlich $a_{k+2} := 0$ setzen, haben wir nämlich

$$\begin{aligned} a_i &= q_i a_{i+1} + a_{i+2} \\ &= q_i(q_{i+1} a_{i+2} + a_{i+3}) + a_{i+2} \\ &= (q_i q_{i+1} + 1) a_{i+2} + q_{i+1} a_{i+3} \\ &> 2a_{i+2}. \end{aligned}$$

Die größte vorkommende Zahl halbiert sich also alle zwei Iterationen. Wenn wir also $2\lceil \log_2(a_1)\rceil$ Iterationen benötigen würden, wäre die größte vorkommende positive natürliche Zahl, die wir dann betrachten kleiner als 1. Aus diesem Widerspruch schließen wir, dass $k < 2\lceil \log_2(a_1)\rceil$ ist. Nehmen wir noch die eine Iteration zur eventuellen Vertauschung der Startziffern hinzu, erhalten wir die Behauptung.

Nun zur Behauptung, dass der euklidische Algorithmus kein streng polynomialer Algorithmus ist. Da die Eingabe stets aus 2 Zahlen besteht, ist dies gleichbedeutend damit, dass die Anzahl der Iterationen des Algorithmus nicht beschränkt ist. Dafür betrachten wir die Folge der Fibonaccizahlen definiert durch

$$F_0 = 0 \qquad F_1 = 1 \qquad F_n = F_{n-1} + F_{n-2} \text{ für } n \geq 2$$

und zeigen mittels vollständiger Induktion über $n \geq 1$:

> Bei Eingabe von F_{n+2} und F_{n+1} benötigt der euklidische Algorithmus n Iterationen.

Für $n = 1$ haben wir $F_3 = 2$ und $F_2 = 1$ und der Algorithmus terminiert nach der ersten Iteration:

Sei also $n > 1$. Dann berechnet der euklidische Algorithmus in der ersten Iteration

$$F_{n+2} = 1 \cdot F_{n+1} + F_n$$

und fährt mit F_{n+1} und F_n fort. Dafür benötigt er nach Induktionsvoraussetzung $n-1$ weitere Schritte, woraus die Behauptung mit dem Induktionsprinzip folgt.

6.7.4 Lösung (zu Aufgabe 6.3.4)**.**

i) Da der zulässige Bereich von (LP_n) offensichtlich beschränkt ist, ist der Linearitätsraum leer und insbesondere hat jede Seitenfläche eine Ecke. Hieraus folgt, dass die Gleichheitsmenge einer Seitenfläche aus einer Teilmenge der Ungleichungen und Vorzeichenrestriktionen bestehen muss, deren Indizes disjunkt sind. Somit ist die Bedingung notwendig.

   Betrachten wir umgekehrt eine Partition der Indizes in Ungleichungen und Vorzeichenrestriktionen, so ist dies nach Satz 6.3.2 die Gleichheitsmenge einer Ecke v. Da die Ecke v Dimension Null hat und das Polyeder Dimension n hat, muss jede der k-elementige Teilmenge der Gleichheitsmenge eine $(n-k)$-dimensionale Seitenfläche definieren. Also ist die Bedingung auch hinreichend.

ii) Nach dem vorherigen Teil entsprechen die Seitenflächen des zulässigen Bereichs von (LP_n) bijektiv, den Vektoren in $\{1, 0, *\}$, wobei 1 in der i-ten Koordinate bedeutet, dass die i-te Ungleichung in der Gleichheitsmenge ist, 0, dass die i-te Vorzeichenrestriktion in der Gleichheitsmenge ist und $*$, dass keine von beiden dazu gehört.

   Der Verband auf diesen Vektoren, der durch die Relationen $0 \leq *$ und $1 \leq *$ induziert wird, ist offensichtlich sowohl zum Seitenflächenverband des Klee-Minty-Cubes (LP_n) als auch zu dem von H_n isomorph. Die Behauptung folgt also aus der Transitivität der Isomorphie.

6.7.5 Lösung (zu Aufgabe 6.3.9)**.**

Wir betrachten ein Tableau zu (LP_n), bei dem allerdings die Spalten so umsortiert

sind, dass Spalten zu echten Variablen und Schlupfvariablen alternieren. Dies ändert offensichtlich nichts an der Gültigkeit von 6.3.1, 6.3.2, 6.3.3 und 6.3.5.

Wir gehen nun vor wie im Beweis von Satz 6.3.6. Die Induktionsverankerung können wir identisch übernehmen. Zu untersuchen bleibt also der Induktionsschritt.

Beachten Sie, dass die Ordnung der Variablen nur für die Pivotauswahl gilt. In der folgenden Darstellung behalten wir, außer im relevanten letzten Index, die Sortierung der Spalten zu echten und Schlupfvariablen aus Gründen der Übersichtlichkeit wie bei der Dantzigschen Gradientenregel bei.

Sei $n > 1$. Nach Induktionsannahme haben das erste und das 2^{n-1}-te Tableau von LP_{n-1} folgende Gestalt

c	0	1	0	0
A	I_{n-2}	0	0	b
$2c$	0	1	1	5^{n-1}

Tableau 1

$-c$	0	0	-1	-5^{n-1}
A	I_{n-2}	0	0	b
$2c$	0	1	1	5^{n-1}

Tableau 2^{n-1}

Wieder betrachten wir Tableaus für das Problem LP_n.

$2c$	0	2	0	1	0	0
A	I_{n-2}	0	0	0	0	b
$2c$	0	1	1	0	0	5^{n-1}
$4c$	0	4	0	1	1	5^n

Tableau 1

$-2c$	0	0	-2	1	0	$-2\cdot 5^{n-1}$
A	I_{n-2}	0	0	0	0	b
$2c$	0	1	1	0	0	5^{n-1}
$-4c$	0	0	-4	1	1	5^{n-1}

Tableau 2^{n-1}

$2c$	0	0	2	0	-1	$-3\cdot 5^{n-1}$
A	I_{n-2}	0	0	0	0	b
$2c$	0	1	1	0	0	5^{n-1}
$-4c$	0	0	-4	1	1	5^{n-1}

Tableau $2^{n-1}+1$

$-2c$	0	0	-2	1	0	5^n
A	I_{n-2}	0	0	0	0	b
$2c$	0	1	1	0	0	5^{n-1}
$-4c$	0	0	-4	1	1	5^{n-1}

Tableau 2^n

Mit der gleichen Argumentation wie bei der Dantzigschen Regel schließen wir, dass die $(2n-1)$-ste Spalte in den ersten $2^{n-1}-1$ Schritten nicht zur Pivotspalte wird, also die letzte Zeile auch nicht zur Pivotzeile. Wiederum ist die Pivotwahl identisch zu der Wahl für LP_{n-1} und wir erreichen Tableau 2^{n-1} und Tableau $2^{n-1}+1$ als das darauf Folgende. Auch dieses sieht wieder aus wie ein erweitertes erstes Tableau für LP_{n-1}, nur dass wieder zwei Spalten vertauscht sind. Dies ändert jedoch nichts an der Pivotwahl, da das Element, was in die optimale Basis gehört,

den größten Index hat. Folglich müssen wir wiederum insgesamt 2^n Iterationen vornehmen.

6.7.6 Lösung (zu Aufgabe 6.4.4)**.**
Wir zeigen dies mittels doppelter vollständiger Induktion über d und n. Wir verankern für $d = 1$ und n beliebig. Die n Punkte auf der Gerade unterteilen diese in $n-1 = \binom{n-1}{1}$ Strecken und zwei Halbstrahlen. Da

$$\binom{n}{0} + \binom{n}{1} = 1 + n = \binom{n-1}{1} + 2,$$

ist die Behauptung auch für den allgemeinen Fall verankert.

Sei nun $d > 1$. Besteht unser Arrangement $\mathscr{H}$ nur aus einer Hyperebene, so haben wir nur zwei unbeschränkte Zellen, also $\binom{0}{d} = 0$ beschränkte und $\binom{1}{0} + \binom{1}{1} = 2$ volldimensionale Zellen. Sei also $n > 1$. Wir betrachten eine feste Hyperebene H. Da die Hyperebenen in allgemeiner Lage sind, also keinerlei Parallelität auftritt, induziert der Schnitt von H mit den übrigen Hyperebenen auf H ein $(d-1)$-dimensionales Hyperebenarrangement. Nach Induktionsvoraussetzung gilt für dieses Arrangement $\mathscr{H}'$

$$L_b(\mathscr{H}') = \binom{n-2}{d-1}, \qquad L(\mathscr{H}') = \sum_{i=0}^{d-1} \binom{n-1}{i}.$$

Außerdem betrachten wir das d-dimensionale Hyperebenenarrangement, $\mathscr{H}''$, das entsteht, wenn wir H aus $\mathscr{H}$ entfernen. Dann gilt nach Induktionsvoraussetzung

$$L_b(\mathscr{H}'') = \binom{n-2}{d}, \qquad L(\mathscr{H}'') = \sum_{i=0}^{d} \binom{n-1}{i}.$$

Wenn wir nun H zu $\mathscr{H}$ wieder hinzufügen, entsteht für jede beschränkte Zelle aus $\mathscr{H}'$ eine zusätzliche beschränkte Zelle. Durchschneidet H eine beschränkte Zelle von $\mathscr{H}''$, so ist dies sofort klar. Durchschneidet sie aber eine unbeschränkte Zelle, so muss, da die Schnittfläche beschränkt ist, eine der beiden entstehenden Zellen beschränkt sein. In beiden Fällen erzeugen wir also genau eine zusätzliche beschränkte Zelle. Offensichtlich erzeugt jede volldimensionale Zelle in $\mathscr{H}'$ genau eine zusätzliche volldimensionale Zelle in $\mathscr{H}$ gegenüber $\mathscr{H}''$.

Also schließen wir

$$\begin{aligned} L_b(\mathscr{H}) &= L_b(\mathscr{H}') + L_b(\mathscr{H}'') \\ &= \binom{n-2}{d-1} + \binom{n-2}{d} = \binom{n-1}{d} \end{aligned}$$

und ebenso

$$\begin{aligned} L(\mathscr{H}) &= L(\mathscr{H}') + L(\mathscr{H}'') \\ &= \sum_{i=0}^{d-1} \binom{n-1}{i} + \sum_{i=0}^{d} \binom{n-1}{i} \\ &= 1 + \sum_{i=1}^{d} \left(\binom{n-1}{i-1} + \binom{n-1}{i} \right) \\ &= \binom{n}{0} + \sum_{i=1}^{d} \binom{n}{i} \\ &= \sum_{i=0}^{d} \binom{n}{i}. \end{aligned}$$