Seminar: Intelligente Algorithmen
Stefan Kopp, Alfred Kranstedt, Nadine Leßmann

Lineare Programmierung

Frank Schönmann

WS 2003/04

Inhaltsverzeichnis

1 Motivation **3**

2 Lineare Programmierung (LP) **4**
2.1 Einführendes Beispiel 4
2.2 Anwendungsbeispiele 5
2.3 Formale Formulierung 5
2.4 Normalformen 6
2.5 Geometrische Interpretation 7
2.6 Spezialfall: diskrete lineare Programmierung 8

3 Methoden zur Lösung von LP-Problemen **9**
3.1 Simplex-Algorithmus 9

4 Lösung des TSP mithilfe von LP **10**

5 Schnittebenen **11**

6 Zusammenfassung **12**

Abbildungsverzeichnis

1 Lösung des *traveling salesman problems* für 15.112 Städte in Deutschland . . . 3
2 Geometrische Interpretation eines linearen Programms 8
3 Lösungsraum eines diskreten linearen Optimierungsproblems 9
4 Alle Verbindungen zwischen vier Städten und eine der gültigen Touren 10
5 Aufteilung einer Menge von Städten in zwei Teilmengen 12

1 Motivation

Bereits 1954 stellten Dantzig, Fulkerson & Johnson eine Methode zur Lösung des TSP vor, die auf linearer Programmierung beruht. Mit dieser Methode lösten sie eine 49-Städte-Instanz, eine beeindruckende Leistung zur damaligen Zeit. Auch der aktuelle Weltrekord, eine optimale Tour durch 15.112 Städte in Deutschland, wurde von Applegate, Bisby, Chvátal & Cook mit einer Weiterentwicklung der *cutting planes*-Methode gelöst.

Abb. 1: Lösung des *traveling salesman problems* für 15.112 Städte in Deutschland

Einführung und Beispiele der vorliegenden Ausarbeitung orientieren sich vor allem an [CLRS01], die Lösungsansätze zur Behandlung des *traveling salesman problems* mithilfe der linearen Programmierung an [ABCC98] und [ABCC01].

2 Lineare Programmierung (LP)

Grundidee der linearen Programmierung ist die Optimierung einer linearen Funktion mit n Freiheitsgraden, die durch lineare Gleichungen und Ungleichungen eingeschränkt ist. Diese Einschränkungen können z. B. widersprüchliche Bedingungen oder beschränkte Ressourcen darstellen.

Die lineare Programmierung ist eines der Hauptverfahren des *Operations Research*, um lineare Optimierungsprobleme zu lösen. Dabei stellt sie eine sehr allgemeine Methode zur Optimierung von Problemlösungen dar, für die keine speziell entwickelten Algorithmen bekannt sind.

2.1 Einführendes Beispiel

Angenommen ein Politiker möchte seine Wahlkampfgelder möglichst effizient einsetzen, d. h. einerseits so wenig wie möglich ausgeben, aber andererseits in der Stadt mit 100.000 Einwohnern, der Vorstadt mit 200.000 Einwohnern und auf dem Land mit 50.000 Einwohnern jeweils eine absolute Mehrheit erringen.

Abhängig davon, in welches Wahlkampfthema er investiert, steigt oder sinkt seine Beliebtheit in den drei Bevölkerungsgruppen, was sich direkt auf seinen Wahlerfolg auswirkt (siehe Tabelle 1).

Wahlkampfthema	Variable	Stadt	Vorstadt	Land
Straßenbau	x_1	-2	5	3
Waffenkontrolle	x_2	8	2	-5
Subventionen	x_3	0	0	10
Mineralölsteuer	x_4	10	0	-2

Tab. 1: Erwartete Gewinne bzw. Verluste (in Tausend) an Wählerstimmen pro \$1000

Die Tabelle ist folgendermaßen zu interpretieren: werden \$1000 in den Wahlkampf für Straßenbau investiert, sinken die erwarteten Wählerstimmen in der Stadt um 2000, nehmen in Vorstadt aber um 5000 und auf dem Land um 3000 zu. Analog für die restlichen drei Zeilen.

Die erwarteten Gewinne bzw. Verluste in der Stadt lassen sich, wenn die vier Variablen $x_1, \ldots, x_4$ belegt sind, zusammenfassen als $\Delta_1 = -2x_1 + 8x_2 + 0x_3 + 10x_4$. Um die Bedingung der absoluten Mehrheit zu erfüllen, unterliegt dieser Wert der (linearen) Einschränkung $\Delta_1 \geq 50$. Ähnliches gilt wieder für die beiden anderen Bevölkerungsgruppen, so dass sich das gesamte Problem formal zusammenfassen lässt:

$$\text{minimiere } x_1 + x_2 + x_3 + x_4$$

unter Einhaltung der Nebenbedingungen für eine Mindestanzahl an Wählerstimmen:

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

$$x_1, x_2, x_3, x_4 \geq 0$$

Das Problem der linearen Programmierung ist es nun, diese Aufgabe zu lösen. Zwar ist es bei einfacheren Problemen wie diesem nicht weiter schwierig, durch Ausprobieren eine gültige Lösung zu finden. Aber bereits bei der Suche nach der optimalen Lösung kommt man damit nicht mehr weiter. Dafür gibt es effiziente Verfahren, die in Kapitel 3 vorgestellt werden.

2.2 Anwendungsbeispiele

Viele Probleme, die sich mit linearer Programmierung lösen lassen, sind im wirtschaftlichen Bereich zu finden. Ein Beispiel dafür ist die effiziente Verteilung von Crews einer Airline auf alle möglichen Flüge unter Einhaltung gesetzlicher Bestimmungen, z. B. der maximalen Arbeitszeit oder der Verteilung auf Flugzeugtypen. Weitere Anwendungsbeispiele finden sich in vielen *Operation Research*-Büchern.

Aber nicht nur wirtschaftliche Optimierungsaufgaben lassen sich mit linearer Programmierung lösen, sondern auch abstraktere Probleme, beispielsweise in der Graphentheorie: das Finden kürzester Pfade, des maximalen Flusses oder eines minimalen Kostenflusses lässt sich unter geeigneter Formulierung ebenfalls lösen. Näheres hierzu findet sich auch in [CLRS01].

2.3 Formale Formulierung

Ziel der linearen Programmierung ist es, eine lineare Funktion in Abhängigkeit von einer Menge linearer Ungleichungen zu optimieren. Sei $c_1, \ldots, c_n \in \mathbb{R}$ und $x_1, \ldots, x_n$ eine Menge von Variablen. Dann nennt man f mit

$$f(x_1, \ldots, x_n) = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n = \sum_{i=1}^{n} c_i x_i$$

eine **lineare Funktion**. Sei $b \in \mathbb{R}$ und f eine lineare Funktion. Dann ist

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

eine **lineare Gleichung** und

$$f(x_1, \ldots, x_n) \leq b \qquad f(x_1, \ldots, x_n) \geq b$$

sind **lineare Ungleichungen**. In der linearen Programmierung sind strikte Ungleichungen nicht zugelassen. Formal ist ein **LP-Problem** das Problem, eine lineare Funktion in Abhängigkeit von einer endlichen Anzahl linearer Ungleichungen zu optimieren. Man bezeichnet das Problem als **lineares Minimierungsproblem** oder **lineares Maximierungsproblem**.

Wie wir in Abschnitt 2.4 zeigen, sind alle linearen Programme äquivalent zu einem LP-Problem in Standardform. Dies ist dann folgendermaßen definiert:

$$\text{maximiere } f(x_1, \ldots, x_n) = \sum_{i=1}^{n} c_i x_i$$

unter Einhaltung der linearen Bedingungen

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

$$x_j \geq 0 \qquad \text{für } j = 1, 2, \ldots, n$$

Dasselbe Problem lässt sich in Matrix-Schreibweise definieren als

$$\text{maximiere } f(\mathbf{x}) = \mathbf{c}^T \mathbf{x}$$

unter den Nebenbedingungen[1]

$$A\mathbf{x} \leq \mathbf{b}$$
$$\mathbf{x} \geq 0$$

Da dieses Problem bereits in einer standardisierten Form vorliegt, kann man es kompakt als $(A, \mathbf{b}, \mathbf{c})$ zusammenfassen. Für die vorkommenden Matrizen und Vektoren gelten die Dimensionen $A \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m$ und $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$.

2.4 Normalformen

Unter einer Normalform eines mathematischen Objekts versteht man in der Mathematik eine Darstellung, die bestimmte vorgegebene Eigenschaften hat und für alle Objekte dieses Typs eindeutig bestimmt werden kann.

Im Bereich der linearen Programmierung gibt es zwei Normalformen, die von Interesse sind: In der **Standardform** sind alle Bedingungen durch Ungleichungen definiert, in der **Slackform** durch Gleichungen. Jedes LP-Problem lässt sich durch geeignete Umformungen in beide Normalformen bringen. Mögliche Abweichungen von der Standardform sind

1. Minimierungsproblem statt Maximierungsproblem,
2. Variablen ohne Nichtnegativitätsbedingung,
3. Gleichheitsbedingungen statt Ungleichheitsbedingungen und
4. Größer-als-Bedingungen statt Kleiner-als-Bedingungen.

Formulierungen von linearen Programmen, in denen diese Fälle auftreten, lassen sich umformen durch einige einfachen Operationen:

1. Negierung des Koeffizienten $\mathbf{c}$ in der Zielfunktion,
2. Ersetzung von x_j durch $x'_j - x''_j$ mit $x'_j, x''_j \geq 0$,
3. Ersetzung von $f(\mathbf{x}) = b$ durch $f(\mathbf{x}) \leq b, f(\mathbf{x}) \geq b$ und
4. Negierung der Koeffizienten a_{ij}, b_i in der betreffenden Bedingung i.

Beispiel Gegeben sei das folgende lineare Optimierungsproblem:

$$\text{minimiere } -2x_1 + 3x_2$$

unter Einhaltung der Nebenbedingungen:

$$\begin{array}{rcrcl} x_1 & + & x_2 & = & 7 \\ x_1 & - & 2x_2 & \leq & 4 \\ x_1 & & & \geq & 0 \end{array}$$

[1] Zu diesem Zweck ist es nötig, die Relationen $\geq$ und $\leq$ geeignet auf Vektoren zu definieren: Seien $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Dann gilt $\mathbf{x} \geq \mathbf{y}$ gdw. $x_1 \geq y_1 \wedge x_2 \geq y_2 \wedge \cdots \wedge x_n \geq y_n$. $\leq$ analog.

Durch schrittweise Anwendung der obigen Regeln lässt sich dieses lineare Programm in Standardform überführen. Man erhält dann das äquivalente Optimierungsproblem:

$$\text{maximiere } 2x_1 - 3x_2 + 3x_3$$

unter folgenden Nebenbedingungen:

$$\begin{array}{rcrcrcr} x_1 & + & x_2 & - & x_3 & \leq & 7 \\ -x_1 & - & x_2 & + & x_3 & \leq & -7 \\ x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 \end{array}$$

$$x_1, x_2, x_3 \geq 0$$

Um ein LP-Problem in Standardform in die Slackform zu überführen, verwendet man die Äquivalenz der folgenden beiden Ausdrücke:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \Longleftrightarrow \qquad 0 \leq s_i = b_i - \sum_{j=1}^{n} a_{ij} x_j$$

Die s_i nennt man Slack-Variablen. Außer den Nichtnegativitätsbedingungen erhält man nur noch Gleichungen als Nebenbedingungen. Die Slackform eines LP-Problems ist besonders für die praktische Anwendung des Simplex-Algorithmus von Bedeutung, die hier nicht näher erläutert wird.

2.5 Geometrische Interpretation

Es gibt eine sehr intuitive geometrische Interpretation von linearen Optimierungsproblemen, die das allgemeine Verständnis verbessern kann. Diese bezieht sich in der folgenden Erläuterung auf lineare Programme in Standardform; aber für äquivalente Darstellungen lässt sich eine analoge Interpretation finden.

Geht man von einer linearen Gleichung über die Variablen $x_1, \ldots, x_n$ aus, so beschreibt diese eine Hyperebene im n-dimensionalen Raum, d. h. alle Punkte $(x_1, \ldots, x_n)^T$, die die entsprechende Gleichung erfüllen, liegen in dieser Hyperebene[2]. Die zugehörige lineare Ungleichung beschreibt dann einen Halbraum, nämlich alle Punkte, die auf der einen Seite der Hyperebene liegen (inkl. der Hyperebene selbst). Dieser Halbraum stellt die Menge aller gültigen Lösungen für die gegebene Ungleichung dar.

Kombiniert man mehrere lineare Ungleichungen zu einem Ungleichungssystem, so erhält man als Lösungsmenge gerade die Punkte, die alle Ungleichungen erfüllen, also den Schnitt der Halbräume. Dieser Schnitt erzeugt immer ein konvexes Polytop (bzw. einen Simplex, nach dem der Simplex-Algorithmus in Abschnitt 3.1 benannt ist)[3].

Die zu maximierende Zielfunktion stellt ebenfalls eine Hyperebene dar, allerdings mit noch unbekanntem Abstand vom Ursprung und damit unbekanntem Zielwert. Lässt man eine Ebene mit der entsprechenden Steigung vom unendlichen auf den Ursprung zuwandern, so ist die Lösung erreicht, sobald die Ebene zum ersten Mal den Simplex berührt.

[2] Eine Hyperebene im n-dimensionalen Raum ist eine $n-1$-dimensionale, lineare Manigfaltigkeit, die den Raum in zwei Hälften spaltet. Im 2-dimensionalen ist das z. B. eine Gerade, im 3-dimensionalen eine Ebene.

[3] Ein geometrischer Körper K ist konvex, wenn für alle Punktepaare (a, b) mit $a, b \in K$ die Verbindungsstrecke zwischen a und b komplett in K liegt.

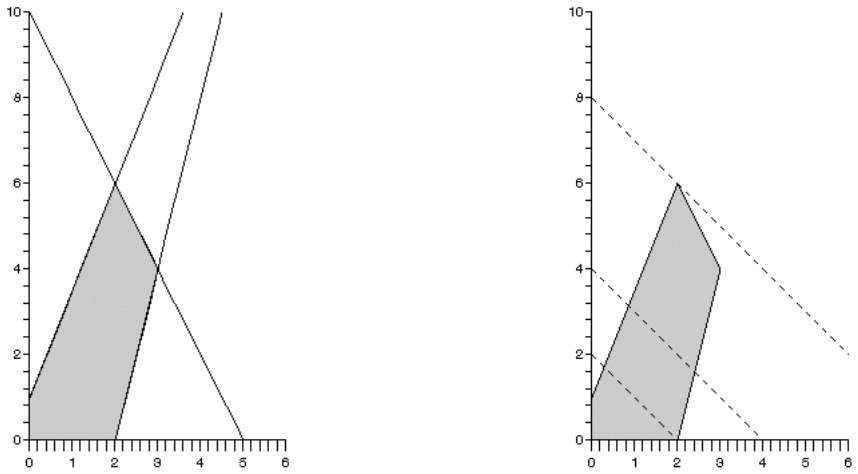
Beispiel Gegeben sei das folgende lineare Optimierungsproblem:

$$\text{maximiere } x_1 + x_2$$

unter Einhaltung folgender Nebenbedingungen:

$$\begin{array}{rcrcr} 4x_1 & - & x_2 & \leq & 8 \\ 2x_1 & + & x_2 & \leq & 10 \\ 5x_1 & - & 2x_2 & \leq & -2 \\ & x_1, x_2 & & \geq & 0 \end{array}$$

In Abbildung 2 ist die geometrische Veranschaulichung dieses Beispiels zu sehen. Die Lösung des Problems ist die obere Ecke des Polygons, die von der Geraden geschnitten wird.



(a) **(b)**

Abb. 2: Geometrische Interpretation eines linearen Programms. Abb. (a) zeigt den konvexen Schnitt dreier Halbräume, die durch entsprechende lineare Ungleichungen definiert sind. Abb. (b) zeigt mögliche Schnitte mit einer Zielfunktion $x_1 + x_2$.

2.6 Spezialfall: diskrete lineare Programmierung

Ein Spezialfall der linearen Programmierung ist die sogenannte diskrete lineare Programmierung (engl. *integer linear programming*). Bei diesen Optimierungsproblemen muss zusätzlich zu den Einschränkungsungleichungen noch gelten $\mathbf{x} \in \mathbb{Z}^n$, d. h. für die Variablen x_i werden nur ganzzahlige Werte zugelassen.

Durch diese zusätzliche Bedingung wird der Raum der Lösungen natürlich weiter eingeschränkt. In Abb. 3 ist das beispielhaft in zwei Dimensionen zu sehen. Dennoch sind die Extremwerte des Polytops noch immer reellwertig. Daher liefern die bekannten Algorithmen für lineare Programme in diesem Fall keine gültige Lösung.

Eine weitere Spezialisierung dieser Art von Problemen ist die binäre diskrete lineare Programmierung (engl. *0-1 integer linear programming*), bei der gelten muss $\mathbf{x} \in \{0,1\}^n$, also die x_i nur die beiden Werte 0 oder 1 annehmen.

Auch das *traveling salesman problem* lässt sich als (binäres) diskretes lineares Programm darstellen, wie wir in Abschnitt 4 sehen werden. Leider ist diese Art von Problemen nicht effizient lösbar, da sie in die Klasse der $\mathcal{NP}$-harten Probleme fällt. Dies scheint anschaulich nicht klar, lässt sich aber leicht durch polynomielle Reduktion auf ein $\mathcal{NP}$-vollständiges Problem beweisen, beispielsweise 3SAT oder KNAPSACK.

3 Methoden zur Lösung von LP-Problemen

Es existieren bereits einige fertige Standardverfahren zur Lösung von linearen Optimierungsproblemen, die in einer Normalform vorliegen. Daher müssen solche Probleme nicht mit einer speziellen Lösung angegangen werden, sondern es genügt, ein entsprechendes lineares Programm zu formulieren und einen passenden Algorithmus darauf auszuführen.

Lange Zeit dachte man, dass die lineare Programmierung nicht der Problemklasse $\mathcal{P}$ angehört, bis der russische Mathematiker L. G. Khachian 1979 einen effizienten Algorithmus zur Lösung solcher Probleme vorstellte – das sogenannte Ellipsoidverfahren. Dies liegt in der Effizienzklasse $O(n^6)$, was allerdings für den praktischen Einsatz nicht geeignet ist.

Ein neueres Verfahren stellte N. K. Karmarkar vor, die *interior point*-Methode mit einer Effizienz von $O(n^{3.5})$, die es von der Geschwindigkeit her mit dem Simplex-Algorithmus aufnehmen kann.

3.1 Simplex-Algorithmus

Der Simplex-Algorithmus ist die in der Praxis am häufigsten eingesetzte Methode zur Lösung von linearen Optimierungsproblemen. Zwar lassen sich Anwendungsfälle konstruieren,

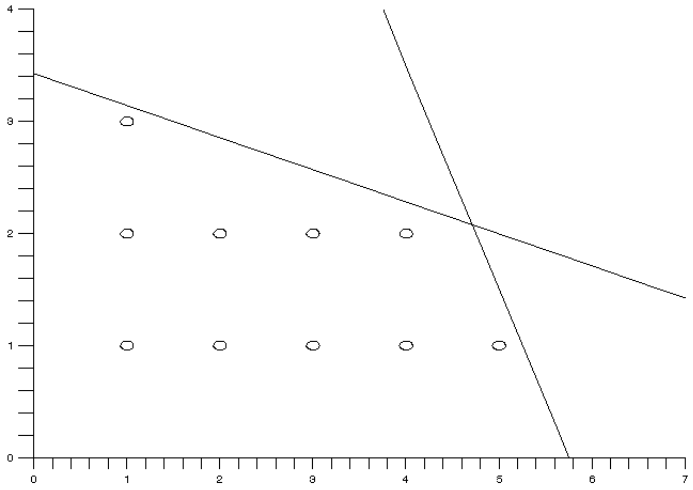


Abb. 3: Lösungsraum eines diskreten linearen Optimierungsproblems sind in diesem Beispiel nur die markierten Punkte. Ein effizientes Finden eines solchen Punktes ist nicht möglich.

in denen dieses Verfahren keine polynomielle Laufzeit mehr aufweist, doch die Mehrzahl aller Aufgaben löst es effizient. Der ursprüngliche Algorithmus wurde 1947 von DANTZIG vorgestellt und im Laufe der Zeit deutlich verbessert.

Wie aus dem Namen abzulesen ist, nutzt der Simplex-Algorithmus die Simplex-Form des Lösungsraums aus, um seine Aufgabe zu bearbeiten. Der Algorithmus erhält als Eingabe ein lineares Programm in Slackform und berechnet dann eine optimale Lösung.

Anschaulich beginnt das Verfahren an einer Ecke des Polytops und führt eine Kette von Iterationen durch, bei denen der Algorithmus jeweils zu einer benachbarten Ecke springt, sofern deren Wert nicht kleiner als der aktuelle ist. Der Wert berechnet sich natürlich durch Einsetzen der Ecke als Lösung in die Zielfunktion.

Der Algorithmus terminiert, sobald er ein lokales Maximum findet, d. h. keine benachbarte Ecke einen höheren Zielwert aufweist. Dieses Maximum ist global wegen der Konvexität des Polytops und der Linearität der Zielfunktion. Der ausführliche Beweis findet sich in [CLRS01].

4 Lösung des TSP mithilfe von LP

Um unsere Betrachtungen durchzuführen, beschreiben wir das *traveling salesman problem* auf formale Weise in einer Art, die bereits an die lineare Programmierung erinnert: Gegeben sei ein TSP mit n Städten, spezifiziert durch einen Kostenvektor $\mathbf{c} \in \mathbb{R}^{n(n-1)/2}$. Jede mögliche Tour sei durch einen Inzidenzvektor $\mathbf{x} \in \mathcal{S} \subset \{0,1\}^{n(n-1)/2}$ dargestellt, wobei $\mathcal{S}$ die Menge der Touren darstellt (alle Hamilton-Zyklen). Der Inzidenzvektor gibt dabei an, ob die zu einer Komponente x_e gehörige Kante in der Tour enthalten ist ($x_e = 1$) oder nicht ($x_e = 0$).

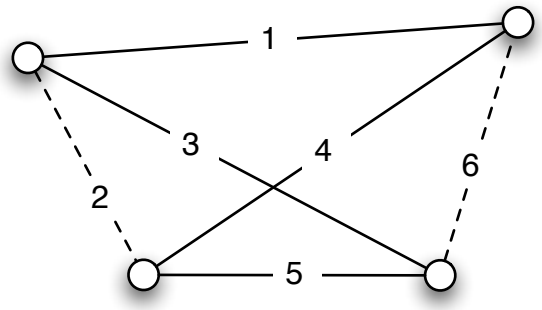


Abb. 4: Alle Verbindungen zwischen vier Städten und eine der gültigen Touren. Der Inzidenzvektor lautet in diesem Fall $\mathbf{x}^T = (1\,0\,1\,1\,1\,0)$ und beschreibt, welche Kanten in der Lösung enthalten sein sollen.

Da $\mathbf{c}^T\mathbf{x}$ genau die Kosten einer Tour $\mathbf{x}$ ergibt, lautet das *traveling salesman problem* dann:

$$\text{minimiere } \mathbf{c}^T\mathbf{x} \text{ mit } \mathbf{x} \in \mathcal{S} \tag{1}$$

Allerdings hat man nun ein Problem, das so nicht ohne weiteres lösbar ist. Daher formuliert man eine *relaxation*, d. h. ein erweitertes Problem, das allerdings einfacher zu behandeln ist:

$$\text{minimiere } \mathbf{c}^T\mathbf{x} \text{ mit } A\mathbf{x} \leq \mathbf{b}, \tag{2}$$

wobei $A\mathbf{x} \leq \mathbf{b}$ ein lineares Ungleichungssystem ist, das von allen $\mathbf{x} \in \mathcal{S}$ erfüllt wird. Das bedeutet, jede gültige Lösung von (1) ist auch eine Lösung für (2), aber nicht umgekehrt. Man erhält ein einfach zu lösendes lineares Problem $(A, \mathbf{b}, \mathbf{c})$.

Natürlich ist die Lösung von (2) nicht direkt geeignet für das ursprüngliche Problem: Einerseits hat der Lösungsraum evtl. eine größere Ausdehnung, da gefordert ist, dass er mindestens alle gültigen Touren mit einschließt. Andererseits wurde die Bedingung der diskreten Variablenbelegungen fallengelassen, um ein effizientes Verfahren anwenden zu können.

Auf zwei Arten kann man die Lösung aber dennoch ausnutzen: man findet zumindest eine untere Schranke für das gegebene *traveling salesman problem*. Eine untere Schranke kann man beispielsweise dazu nutzen, den ermittelten Wert eines Näherungsverfahrens qualitativ einzuschätzen – liegt er nahe an der Schranke, genügt er vielleicht schon und man muss nicht weitersuchen. Außerdem kann man mit einer Lösung von (2) in der Nähe davon nach einer Lösung für (1) suchen.

Zuerst ist es nötig, ein initales LP-Problem aufzustellen, das die Bedingungen der *relaxation* erfüllt. Da der Inzidenzvektor $\mathbf{x}$ nur die Werte 0 und 1 annehmen kann, soll gelten:

$$0 \leq x_e \leq 1$$

Zu beachten ist, dass so auch nicht-diskrete Werte zugelassen werden, d. h. eine bestimmte Kante ist zu einem Bruchteil zwischen 0 und 1 in einer Tour enthalten, was anschaulich natürlich keinen Sinn macht.

Als weitere einschränkende Bedingung berücksichtigt man, dass jede Stadt v auf einer Rundreise nur von genau zwei Kanten besucht wird:

$$\sum \{x_e \mid v \in e\} = 2$$

Diese Bedingungen werden von allen $\mathbf{x} \in \mathcal{S}$ erfüllt und können daher als initiales LP-Problem verwendet werden. Um jetzt zu einer Lösung für das eigentliche Problem (1) zu kommen, muss man den Lösungsraum Schritt für Schritt einschränken. Dies geschieht mit Hilfe von *cutting planes*, die im nächsten Abschnitt beschrieben werden.

5 Schnittebenen

Hat das erweiterte Problem (2) eine Lösung und das Polytop $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ einen Extrempunkt, dann findet ein LP-Algorithmus (z. B. der Simplex-Algorithmus) diesen Punkt als optimale Lösung $\mathbf{x}^*$. Jetzt gilt es zwei Fälle zu betrachten: ist $\mathbf{x}^* \in \mathcal{S}$, so hat man die optimale Lösung für das zugehörige TSP gefunden. Andernfalls existiert wegen der Konvexität des Lösungsraums eine lineare Ungleichung, die von allen Punkten in $\mathcal{S}$ erfüllt wird, aber von $\mathbf{x}^*$ nicht. Man nennt diese Ungleichung **Schnittebene** (engl. *cutting plane* oder *cut*) und fügt sie zum Ungleichungssystem $A\mathbf{x} \leq \mathbf{b}$ hinzu. Dieser Schritt wird wiederholt bis zur Entdeckung einer gültigen Lösung.

Die größte Schwierigkeit hierbei ist das automatische Finden der Schnittebenen bis zur optimalen Lösung für das TSP. Hierfür existieren verschiedene Methoden. 1958 stellte Gomory die nach ihm benannten *Gomory cuts* vor, die aber aus Effizienzgründen heute nicht mehr verwendet werden. Stattdessen werden vor allen Dingen Hypergraph-Schnitte[4] verwendet. Beispiele dafür sind *subtour inequalities*, *blossom inequalities* und *comb inequalities*, die in [ABCC98] und [ABCC01] detaillierter vorgestellt werden.

[4] Ein Hypergraph ist die Verallgemeinerung eines Graphen mit Kanten über beliebig viele Knoten.

Subtour inequalities Dies sind die intuitivsten Ungleichungen, die automatisch zu bestimmen sind. Ihnen zugrunde liegt die Tatsache, dass eine gültige Tour jede Aufteilung von V in zwei echte Teilmengen S und T die beiden Teilmengen durch mindestens zwei Kanten verbinden muss. Dies lässt sich leicht einsehen, da der Handlungsreisende beim Wechsel von einer Stadt aus S in eine Stadt aus T wieder in eine der Städte aus S zurückkehren muss, um seine Tour zu vollenden. Veranschaulicht ist dies in Abbildung 5.

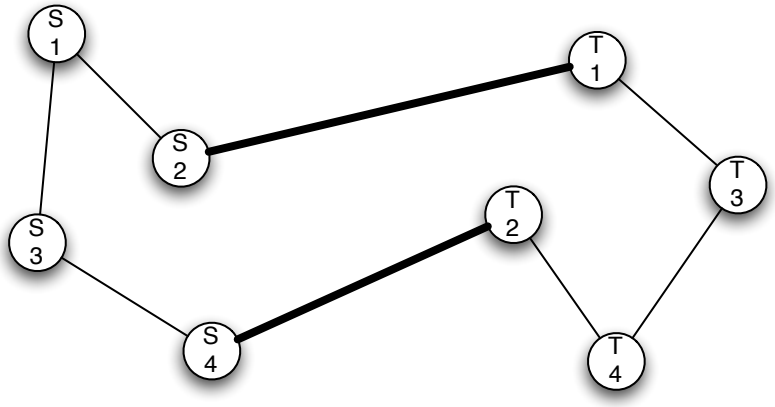


Abb. 5: Die Menge der Städte wurde hier aufgeteilt in zwei echte Teilmengen S und T. Fett markiert sind die beiden Verbindungskanten zwischen diesen Teilmengen. Jede gültige Tour enthält mindestens zwei solcher Verbindungskanten.

Die Zahl der Verbindungskanten zwischen den Knotenmengen S und T ist definiert als

$$x(S,T) = \sum\{x_e \mid e \cap S \neq \emptyset \neq e \cap T\},$$

denn gezählt wird jede vorhandene Kante, die sowohl Knoten aus S als auch Knoten aus T enthält. Die *subtour*-Bedingung lässt sich dann formal angeben als $x(S, V - S) \geq 2$.

In einem iterativen Verfahren wird das anfängliche Ungleichungssystem dann jeweils um eine deratige Ungleichung erweitert, falls eine *subtour* in der gefundenen Lösung des erweiterten Problems auftritt. Es gibt effiziente Methoden, in der erneuten Berechnung einer Lösung mit einem erweiterten Ungleichungssystem die Ergebnisse vorheriger Berechnung einzubeziehen.

6 Zusammenfassung

Lineare Programmierung ist eines der Standardverfahren zur exakten Lösung größerer Instanzen des *traveling salesman problem*. Man kann diese Aufgabe als Problem der binären diskreten linearen Programmierung auffassen. Diese Gruppe von Problemen ist $\mathcal{NP}$-hart und daher nicht effizient lösbar. Zur Lösung kommt man, indem man effiziente Verfahren der einfachen linearen Programmierung iterativ anwendet, während man ein hochdimensionales Polytop, das den Lösungsraum darstellt, mit geeigneten *cut*-Bedingungen immer weiter beschneidet.

Literatur

[ABCC98] David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. On the solution of traveling salesman problems. *Documenta Mathematica*, Extra Volume

Proceedings ICM III (1998):645–656, 1998.

[ABCC01] David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. TSP cuts which do not conform to the template paradigm. *Lecture Notes in Computer Science*, 2241:261–305, 2001.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*, chapter Linear Programming, pages 770–821. The MIT Press, Cambridge, Massachusetts, second edition, 2001.

[GP99] Martin Grötschel and Manfred W. Padberg. Die optimierte Odyssee. *Spektrum der Wissenschaft, Digest*, 2:32–41, 1999.