Wie findet man den optimalen Weg zum Ziel?

Klassische Probleme der Kombinatorischen Optimierung

Teilnehmer/innen:
Markus Dahinten, Graf-Münster-Gymnasium Bayreuth
Robert Fay, Herder-Gymnasium Berlin
Falko Heese, Herder-Gymnasium Berlin
Christian Hofmann, Graf-Münster-Gymnasium Bayreuth
Lena Kalleske, Heinrich-Hertz-Gymnasium Berlin
Irene Winkler, Heinrich-Hertz-Gymnasium Berlin

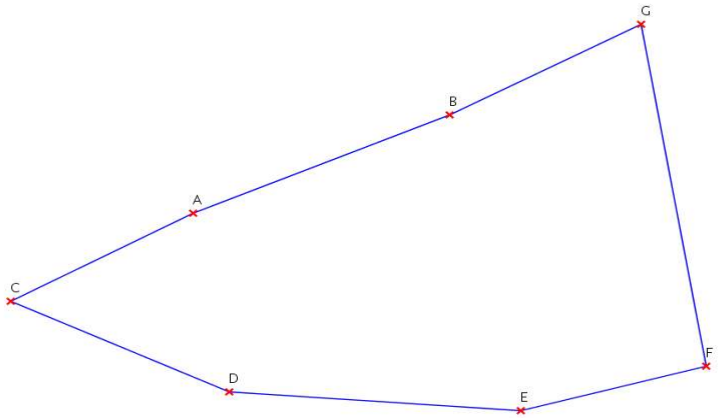
Leitung:
Brigitte Lutz-Westphal, TU Berlin/ZIB

In diesem Kurs befassen wir uns mit den klassischen Problemen der kombinatorischen Optimierung, beginnend beim Travelling-Salesman-Problem. Wir suchen Heuristiken zur Lösung des Problems. Da die optimale Lösung jedoch allgemein nicht erreicht wird, bestimmen wir untere Schranken. Um diese zu finden, betrachten wir aufspannende Bäume.

Das Travelling – Salesman – Problem

Als Ausgang liegt eine Entfernungstabelle mit n Städten vor.

Gesucht ist eine kürzeste Rundreise durch alle Orte, wobei jeder Ort genau einmal besucht wird.



Definition eines Graphen:

Ein Graph ist ein Gebilde aus Knoten und Kanten, wobei Kanten stets in Knoten enden.

Streckenlängen werden als Kantengewichte angegeben.

Graphen können als Adjazenematritzen dargestellt werden.

Lösung:

Vorschlag 1: „Nächster Nachbar“(Konstruktions-Heuristik)

1. Man wählt die kürzeste Strecke der Tabelle, und verbindet von einem der beiden Anschlussknoten zum nächst entferntesten Knoten.
2. Man geht von diesem Knoten zum nächst entferntesten noch nicht benutzten Knoten.
3. Wiederhole 2. vom neuen Knoten aus bis kein nicht benutzter Knoten mehr existiert.
4. Kehre zum Ausgangsknoten zurück.

Diese Heuristik hat den Vorteil, dass sie schnell und einfach ist. Sie hat jedoch den Nachteil, dass sie nicht sehr gute Ergebnisse liefert, da oft lange Strecken zum Ende entstehen.

Vorschlag 2: „Doppelter Nächster Nachbar“ (Konstruktions-Heuristik)

Hierbei geht man ähnlich der ersten Methode vor, mit dem Unterschied, dass man nun jeweils mit der nächsten kürzesten Entfernung eines der beiden Anschlussknoten fortfährt.

Vorschlag 3: „B illigste Insertion“ (Verbesserungs-Heuristik)

Die Billigste Insertion ist ein Verbesserungsverfahren, bei dem man aus einer fertigen Rundreise einen Knoten herausnimmt und dessen Nachbarn miteinander verbindet. Anschließend berechnet man die neuen Gesamtdistanzen für jede mögliche neue Position dieses Knoten in der gesamten Rundreise. Der Knoten wird dort wieder eingefügt, wo die Differenz zwischen der Summe der Gewichte der beiden neu angeschlossenen Kanten und des Kantengewichts der bisherigen Verbindung am kleinsten ist.

Vorschlag 4: (Konstruktions-Heuristik)

1. Man betrachtet Knoten mit der größten Zeilensumme, welche noch nicht zwei Markierungen erhalten hat.
2. Von diesem Knoten aus werden die maximal zwei kürzesten verfügbaren Verbindungen ausgesucht.
   (Als verfügbar gelten Knoten, die eine oder keine Markierung haben, wobei sofern mindestens ein nichtmarkierter Knoten existiert stets ein markierter und ein nichtmarkierter Knoten zu wählen sind.
3. Die Endknoten jeder neuen Kante werden markiert.
4. zurück zu Punkt 1, außer es sind keine weiteren Knoten verfügbar

Enumeration

Die einzige Lösung, um ein Optimum zu erreichen, wäre alle $(n-1)!/2$ (n:= Anzahl der Knoten) Möglichkeiten auszuprobieren. Da sich die Anzahl an Möglichkeiten für jeden neuen Knoten vervielfacht, ist diese Lösung für große n für den Computer nicht in praktikabler Zeit zu bewältigen. Z.B. bräuchte ein Rechner für 25 Knoten etwa 10 Millionen Jahre.

Charakterisierung einer Rundreise:
Jeder Knoten muss Grad 2 haben und für jede Teilmenge von n Knoten muss gelten: Es müssen mindestens 2 Kanten aus dieser Teilmenge herausgehen. (Ansonsten wäre es ab 6 Knoten möglich mehrere Kreise zu bilden, die keine Rundreise bilden.)

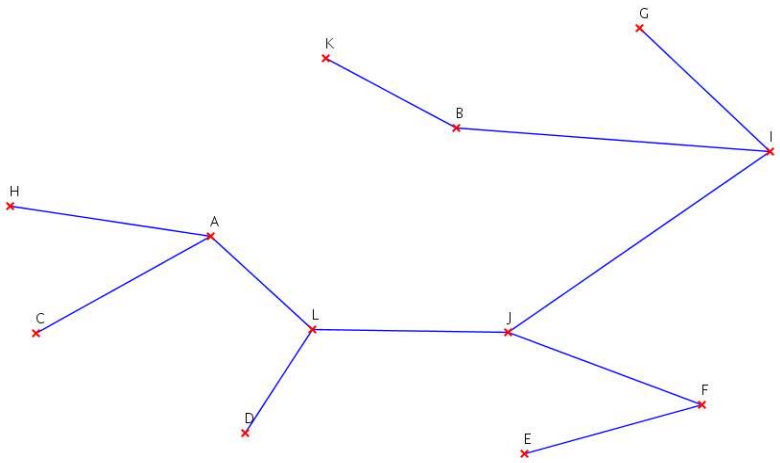
Untere Schranken

Da man heutzutage keinen Algorithmus außer der Enumeration kennt, der sicher das Optimum einer Rundreise errechnet und die Enumeration wegen des zu großen Rechenaufwandes nicht praktisch durchführbar ist, sucht man eine untere Schranke um zu erfahren, wie weit bestimmte Heuristiken vom Optimum entfernt sind.

Hilfsmittel dazu werden im folgenden behandelt. Wir betrachten vorerst das Telefonproblem:
Wir wollen eine Telefongesellschaft gründen und dafür Leitungen bei der Telekom mieten.
Welche Leitungen miete ich damit es möglichst kostengünstig bleibt?

- alle Knoten sollen verbunden sein (zusammenhängend)
- je zwei Knoten sind nur durch genau einen Streckenzug verbunden (keine Kreise)

Dies ist ein **aufspannender Baum**.
Wir suchen also einen minimal aufspannenden Baum, um möglichst billig zu sein. Hierfür gibt es zwei Algorithmen, denen es auch gelingt, das Optimum zu finden, wie wir später zeigen werden.



1. Algorithmus von Kruskal

Definition „Gruppe“: Eine Gruppe ist eine Liste, die die während der Durchführung miteinander verbundenen Knoten zusammenfasst. Die Gruppen sind inkrementell beginnend mit der Zahl 1 benannt.

- Wähle die kürzeste Kante aus der Matrix
- Ist mindestens einer der Knoten in einer Gruppe zusammenfasst?
  - Ja: Sind die beiden Knoten in der gleichen Gruppe?
    - Wenn nicht, verbinde die Kanten und weise alle Knoten in allen beteiligten Gruppen der niedrigsten Gruppe zu
  - Nein: Verbinde die Knoten und weise beide Knoten einer neuen Gruppe zu
- Lösche die benutze Kante aus der Matrix
- Zurück zu Punkt 1, solange nicht alle Knoten in der gleichen Gruppe sind

2. Algorithmus von Prim

- Ein beliebiger Knoten wird als Startwert gewählt und in eine Liste geschrieben.
- Von dort aus wählen wir die Kante mit dem geringsten Gewicht als Anbindung zum nächsten Knoten und tragen diesen ebenfalls in die Liste ein.
- Alle Knoten aus der Liste werden jetzt wieder auf ihre geringsten Kantengewichte überprüft, wobei Kanten, die aus 2 Knoten der Liste bestehen nicht beachtet werden dürfen.
- Die geringsten Kantengewichte werden nun nach den bestehenden Regeln angebunden

Um zu zeigen, dass diese zwei Algorithmen einen minimal aufspannenden Baum erzeugen, beweisen wir folgenden Satz:

Satz: Es sei (G,c) ein Graph G mit der Gewichtsfunktion c und T ein aufspannender Baum in G, dann sind äquivalent:

a) T ist optimal

b) Sei e eine „Nichtbaumkante“ zwischen den Knoten x und y, dann gibt es keine Kante auf dem Weg von x nach y auf dem Baum T die teurer ist als e.

c) Sei e eine Baumkante und sei C eine Zusammenhangskomponente des Waldes T- e, dann ist e die billigste Kante aus dem Schnitt der Knotenmenge von C (Menge aller Kanten die je einen Endknoten in und außerhalb von C haben)

Beweis:

(a)=>(b) indirekter Beweis:

Angenommen b sei nicht erfüllt und e ist eine Kante x->y, welche Nichtbaumkante von T sei.
f ist eine Baumkante auf dem Pfad x->y , wobei das Gewicht von f größer als das Gewicht von e sei.
Dann ist (T-f)+e ein aufspannender Baum mit geringerem Gewicht als T.
Dieses steht im Widerspruch zu a und somit folgt (b) aus (a).

(b)=>(c)

Nehme an (c) ist nicht erfüllt: e sei eine Kante aus dem Baum T, und C eine Zusammenhangskomponente des Waldes T- e. Es gibt ein Knoten x innerhalb von C und einen Knoten y außerhalb von C, f sei eine Kante zwischen x und y. Wenn f < e muss im Weg von x nach y eine Kante enthalten sein, die kleiner als e ist. Dieses steht im Widerspruch zu (b) und somit folgt (c) aus (b).

(c)=>(a)

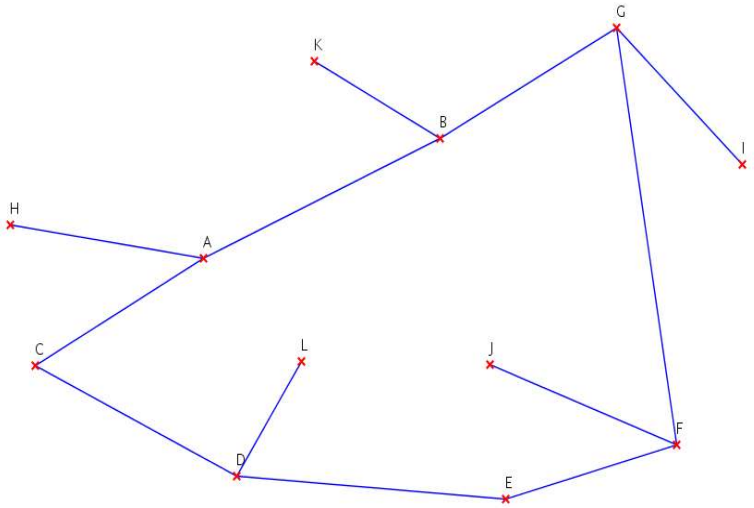
Der Baum T erfülle (c) und der Baum T* erfülle (a), wobei die Schnittmenge aus E(T*) und E(T) maximal sei (d.h. T* und T besitzen maximal viele gemeinsame Kanten). (1)
Wir werden zeigen, dass T*=T. Wir nehmen an, es gäbe eine Kante e zwischen x und y die in T und nicht in T* enthalten ist. Sei C eine Zusammenhangskomponente des Waldes T-e. Da e eine Nichtbaumkante von T* ist enthält der Graph T*+e einen Kreis D (da jeder Baum maximal kreisfrei ist.) Da e im Kreis D und des Schnittes der Knotenmenge von C enthalten ist, gibt es mindestens eine weitere Kante f (f<>e) von D, die im Schnitt der Knotenmenge in C enthalten ist. Wir bemerken, dass (T*+e)-f ein aufspannender Baum ist. Da T* ein minimal aufspannender Baum ist, gilt: c(e) >= c(f). Da wir aber mit (c) für T auch c(e) <= c(f) haben, gilt: c(e) = c(f), und (T*+e)-f ist eine weiterer minimal aufspannender Baum. Dies ist ein Widerspruch zu (1), da (T*+e)-f eine weitere gemeinsame Kante mit T besitzt (nämlich e).
Damit ist die Annahme falsch. Es gibt also keine Kante e, die in T aber nicht in T* enthalten ist. Es gilt also T*=T. Jeder Baum der (c) erfüllt, ist ein minimal aufspannender Baum.

q.e.d.

Der Algorithmus von Kruskal konstruiert einen aufspannenden Baum T. Da durch die Konstruktionsvorschrift immer (b) erfüllt ist, ist T nach dem oben bewiesenen Satz ein minimal aufspannender Baum.

Die Korrektheit des Algorithmus von Prim ist sichergestellt, da nach Konstruktionsvorschrift immer (c) erfüllt ist.

Um jetzt die untere Schranke für Rundreisen zu bestimmen führen wir eine Relaxation des Problems durch (wir lockern die Bedingungen).
Wir definieren einen 1-Baum: Dies ist ein zusammenhängender Graph, der aus einem aufspannenden Baum von (n-1) Knoten und einem Knoten mit 2 Kanten, der an den Baum angebunden wird, besteht.



Es gilt: Jede Rundreise ist ein 1-Baum. Die Menge der Rundreisen durch alle Knoten ist also eine Teilmenge der Menge der 1-Bäume.

Wir suchen minimale 1-Bäume:

- wähle einen beliebigen Knoten
- Konstruiere minimal aufspannenden Baum für die restlichen Knoten
- binde den Knoten k möglichst „billig“a n.

Wie oben festgestellt, sind alle Rundreisen 1-Bäume.
Da die 1-Bäume für jeden Knoten einer Rundreise größer oder gleich der minimalen 1-Bäume der gleichen Knoten sind, ist die optimale Rundreise größer oder gleich aller minimalen 1-Bäume und damit größer gleich dem größten minimalen 1-Baum.

Weitere Betrachtungen an Bäumen:

Ein Baum ist ein kreisfreier zusammenhängender Graph. Ein Wald ist ein kreisfreier Graph, seine Zusammenhangskomponenten sind Bäume.

Satz: Für Bäume besteht folgender Zusammenhang zwischen der Anzahl m der Kanten und n der Knoten: m=n-1.

Beweis:

1.) Es gibt einen Knoten vom Grad 1: Sei P : u, u(1), u(2),... v ein längster Weg, also sind alle Nachbarn von u in P enthalten. U hat also den Grad 1

2.) Vollständige Induktion:

Für n=1 ist die Aussage erfüllt.

Ist die Anzahl der Knoten echt größer als eins, so gibt es einen Knoten mit Grad 1. Um ihn zu finden, verfolgt man, von einem beliebigen Knoten aus startend, die Kanten des Baumes, ohne eine Kante zweimal zu verwenden. Da es keine geschlossenen Wege gibt, erreicht man keine Knoten zweimal und der Weg endet an einem Knoten mit Grad 1. Entfernt man diesen Knoten und die dazugehörige Kante aus dem Graphen, so kann man die Induktionsvorraussetzung anwenden, und somit gilt: m-1= n-2. Nach n-1 Wiederholungen erhält man den Baum mit 1 Knoten und 0 Kanten (nach Induktionsvoraussetzung).

Für einen Wald, der aus z Zusammenhangskomponenten besteht, gilt: m= n-z . Diese Gleichung ergibt sich unmittelbar durch Addition der entsprechenden Gleichungen der einzelnen Zusammenhangskomponenten.

Weiterhin gilt: Die Summe aller Knotengrade in einem Graphen ist zweimal die Anzahl der Knoten. Im Baum mit n Kanten ist die Summe aller Knotengrade 2n-2.
Der Beweis ist trivial.

Wieviele verschiedene aufspannende Bäume gibt es auf n durchnummerierten Knoten?

n=1 => a =1

n=2 => a =1

n=3 => a =3

n=4 => a = 4*4 =16

n=5 => a = 5*25 = 125

Vermutung: a= n^{n-2}

Beweis:

Anzahl der n-2-Tupel $(a_1,a_2,a_3,...a_{n-2})$, a_i € {1,..,n}: n^{n-2}

Zu zeigen: Es gibt eine eineindeutige Zuordnung(Bijektion) zwischen der Menge der aufspannenden Bäume und der n-2-Tupel.

1. Zuordnung von der Menge der Bäume T zur Menge der n-2 Tupel.

   Vorschrift:

   1. Suche den Knoten mit der kleinsten Nummer mit Grad 1, entferne diese Knoten mit Kante aus dem Baum und notiere die Nummer des Nachbarn.
   2. wiederhole 1. bis n-2 Nummern notiert wurden.

   Jedem Baum kann man also eindeutig ein Tupel zuordnen. Jedes Tupel kann erzeugt werden, da man sich ja jeden beliebigen Baum vorstellen kann.

2. Zuordnung von der Menge der n-2 Tupel zur Menge der Bäume T.

   Vorschrift:

   1. Suche das minimale b_1, welches nicht in der Folge $(a_1,..a_{n-2})$ auftritt; dies ergibt die Kante b_1a_1
   2. Suche das minimale b_2<> b_1, welches nicht in der Folge $(a_2,...a_{n-2})$ erscheint

   usf.

Damit kann man die Menge der Bäume der Menge der n-2 Tupel eineindeutig zuordnen. Diese beiden Menge besitzen also die gleiche Mächtigkeit. Es gibt also n^{n-2} aufspannende Bäume.