Proseminar Online – Algorithmen, Prof. Dr. Rolf Klein

Vortrag von Michael Daumen am 13.12.2000

Thema :

Minimum Spanning Tree und 2-Approximation der TSP-Tour

Inhalt des Vortrags :

1. genaue Vorstellung des Problems.
2. Überlegung eines Lösungsansatzes für das Problem des Handlungsreisenden.
3. Definition und Konstruktion des MST zur Approximation der TSP-Tour.
4. 2-Approximation der TSP-Tour mit Hilfe des MST

Im Folgenden werden die Begriffe
TSP für Traveling Salesman Problem oder deutsch Problem des Handlungsreisenden sowie
MST für Minimum Spanning Tree oder deutsch Minimaler Spannbaum benutzt

1. genaue Vorstellung des Problems

Das Problem des Handlungsreisenden

Ein Handlungsreisender soll nacheinander n Städte besuchen. Dabei soll jede Stadt nur genau einmal besucht werden. Am Ende der Reise soll er wieder am Ausgangspunkt ankommen. Gesucht ist dabei eine kürzeste mögliche Route.

Das Problem hat hohe praktische Relevanz. Im Prinzip hat sich jeder von uns schon einmal damit auseinandergesetzt.

Ein Beispiel :

Ein Lkw-Fahrer fährt morgens von seiner Arbeitsstätte los. Dabei soll er 4 Kunden in 4 Städten beliefern und am Schluß wieder zur Arbeitsstätte zurückkehren.

Sein Problem :

Wie lautet die kürzeste Strecke, falls ich jeden Kunden genau einmal besuche ?

Formal kann das TSP mit Hilfe eines *ungerichteten Graphen* ausgedrückt werden :

Die *Knoten* entsprechen den Städten.

Die *Kanten* entsprechen den Verbindungen zwischen den Städten.

Die *Kantengewichte* können Entfernung, Reisezeit etc. entsprechen.

Dabei muß der Graph mindestens einen *einfachen Zyklus* beinhalten, der alle Knoten enthält.

Für einen vollständigen Graphen G mit einer Knotenmenge V = { Bonn, Düsseldorf, Gießen, Koblenz, Köln }ist z.B. Bonn–Köln–Düsseldorf-Giessen–Koblenz–Bonn ein einfacher Zyklus, der alle Knoten enthält.

Definition :

Ein einfacher Zyklus in einem Graphen G, der jeden Knoten von G enthält, wird auch als *Hamiltonscher Zyklus* bezeichnet.

- ➔ Damit ist das TSP äquivalent mit der Suche nach dem kürzesten Hamiltonschen Zyklus in einem ungerichteten Graphen.
- ➔ Der Startknoten ist gegeben und muß dem Endknoten entsprechen.

2. Überlegung eines Lösungsansatzes für das TSP

„Naive“ Methodik : Alle Permutationen testen

- ➔ Führt bei kleineren, dünn besetzten Graphen vielleicht zu Erfolg in vertretbarer Zeit.
- ➔ Doch bei komplexeren, vollständig besetzten Graphen mit großer Knotenmenge praktisch nicht anwendtbar – Laufzeit O (n!).

Satz ohne Beweis :

Das TSP ist NP-vollständig, d.h. jeder z.Z. bekannte Algorithmus hat eine exponentielle Laufzeit von mindestens $O(2^n)$.

Die Aufgabe in der Praxis lautet deshalb : Bestimme eine *möglichst gute Näherungslösung*. Wir werden im Folgenden eine Strategie zur Approximation der TSP-Tour mit Hilfe eines *Minimum Spanning Tree* (MST) vorstellen.

3. Definition und Konstruktion des Minimum Spanning Trees

Def. Spannender Baum, Minimum Spanning Tree

Ein *spannender Baum* eines zusammenhängenden Graphen G ist ein Teilgraph von G, der alle Knoten von G enthält.

Ein *Minimum Spanning Tree* (MST) eines zusammenhängenden Graphen V ist ein spannender Baum von V, dessen Summe der Kantenkosten minimal ist.

Ein MST zu einem vorgegebenen Graphen G läßt sich wahlweise mit *Kruskals Algorithmus* oder dem *Algorithmus von Prim* konstruieren.

Diese zwei Algorithmen zur Konstruktion eines MST zu vorgegebenem zusammenhängendem Graphen G werden nun vorgestellt.

Kruskals Algorithmus :

1 Initialisiere einen Wald mit n isolierten Knoten
2 Initialisiere einen Heap mit allen Kanten
3 REPEAT
4 entferne die billigste Kante (x,y) aus dem Heap

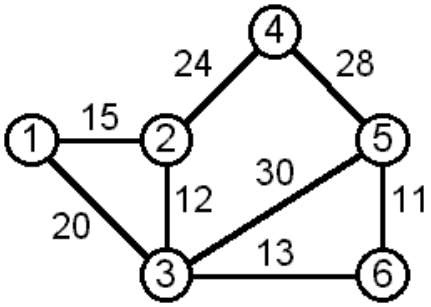
5 falls durch Einfügen von (x,y) zwei Zusammenhangskomponenten verbunden werden, füge (x,y) in den Wald ein

6 UNTIL der Wald besteht nur aus einem Baum

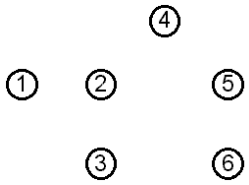
|E| sei die Anzahl der Kanten und |V| die Anzahl der Knoten des Graphen.

Laufzeit von Kruskals Algorithmus : O (|E| * log |E|)

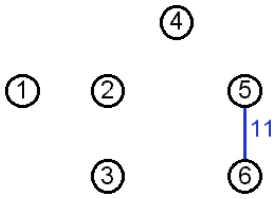
Beispiel zu Kruskals Algorithmus :



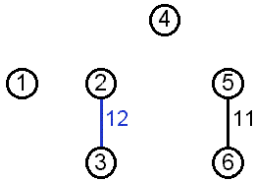
1



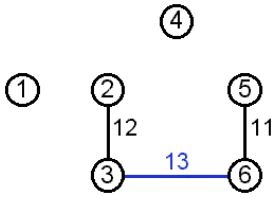
2



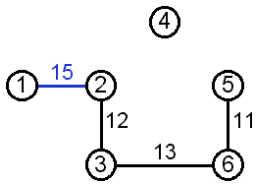
3



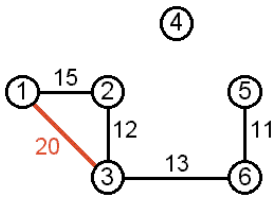
4



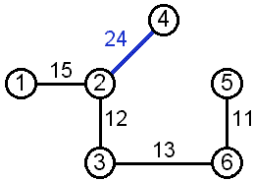
5



6



7



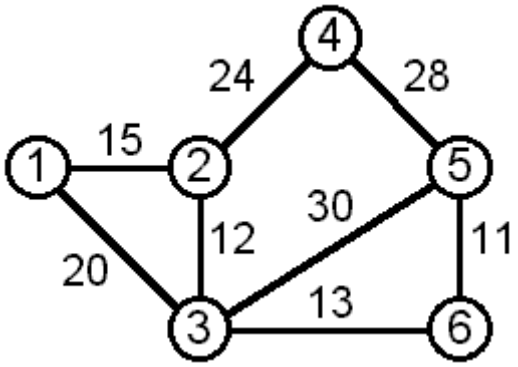
Algorithmus von Prim :

1 Starte mit leerer Kantenmenge
2 Starte mit erstem Knoten: Knotenmenge U = {1}
3 REPEAT
4 Suche Knoten v aus V \ U mit geringstem Abstand zu einem u aus U
5 Nimm Kante (u,v) auf
6 Nimm Knoten v in Knotenmenge U auf
7 UNTIL V \ U ist { }

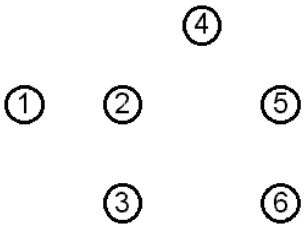
Sei n die Anzahl der Knoten des Graphen. Laufzeit des Algorithmus von Prim : $O(n^2)$

Kruskal benötigt im worst-case $O(n * \log n)$ Rechenschritte bei einem vollständig besetzten Graphen G mit (n über 2) Kanten.
Prim kommt immer mit $O(n^2)$ Schritten aus.

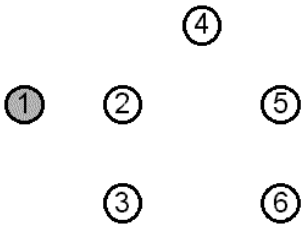
Beispiel zu Prims Algorithmus :



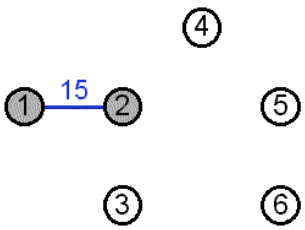
1



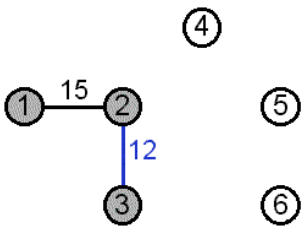
2



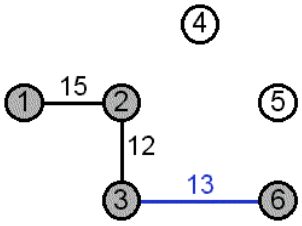
3



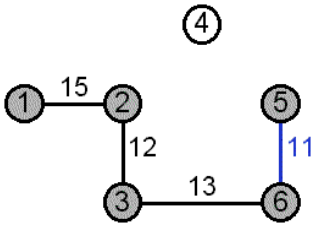
4



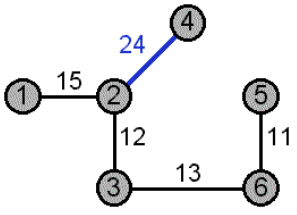
5



6



7



4. 2-Approximation der TSP-Tour mit Hilfe des MST:

Mit Hilfe eines MST läßt sich das TSP folgendermaßen approximieren:
Der Handlungsreisende bewegt sich außen um den MST herum. Da ein MST alle Knoten des zugehörigen Graphen beinhaltet und bei diesem Rundweg um den MST jeder Knoten besucht wird, eignet sich dieses Verfahren zur Approximation der TSP – Tour.
Dieses Verfahren liefert uns gar keine so schlechte Approximation der TSP-Tour und führt zu folgendem Theorem :

Theorem 1 :

Die TSP-Tour um einen minimalen Spannbaum ist weniger als doppelt so lang wie eine optimale TSP-Tour, d.h. sie approximiert die optimale TSP-Tour mit 2.

Diese Aussage wollen wir im Folgenden beweisen.

Beweis :

geg. Ein Graph G
eine optimale TSP-Tour T*
unsere approximierte TSP-Tour T
ein minimaler Spannbaum MST

Die Länge werde im Folgenden jeweils mit c (T*), c (T), c (MST) bezeichnet.

So läßt sich die Korrektheit des Theorems beweisen :

Lasse bei T* eine Kante k zwischen zwei Punkten aus dem Graphen G fort.

→ Dadurch entsteht ein Spannbaum des Graphen G.

Dieser Spannbaum kann aber nach Definition des MST nie kürzer als dieser sein.

Also gilt :	c (MST) <= c (T*) \ k < c (T*)
und natürlich :	**c (MST) < c (T*) [1]**

Der Rundweg um den Spannbaum besitzt die Länge 2 * c (MST), da jede Kante genau 2x traversiert wird.

Wir erhalten :	2 * c (MST) = c (T)
Umformen ergibt :	**c (MST) = ½ * c (T) [2]**

Einsetzen von [2] in [1] liefert :

½ c (T) < c (T*) ⇔ c (T) < 2 * c (T*) q. e. d.

Damit ist die Richtigkeit des obigen Theorems bewiesen.

Zum Abschluss :

Es gibt natürlich noch andere Approximationsverfahren. Genauere Approximationen gehen aber in der Regel zu Lasten der Laufzeit. Die TSP-Approximation zieht nach wie vor die Aufmerksamkeit der Forscher auf sich. Immer genauere Approximationen sollen gefunden werden.Wer mehr über andere Approximationsverfahren wissen möchte, möge sich in der Literatur kundig machen.

Quellenangaben :

Cormen, Leiserson, Rivest An Introduction To Algorithms

Kapitel 24 +37.2 MIT Press 1999

Rolf Klein	Algorithmische Geometrie
S. 225-228	Addison Wesley 1997
F. Preparata	Computational Geometry
Kapitel 6.1	Springer Verlag 1985
E. Horowitz, S. Sahni	Algorithmen
Kapitel 4.6	Springer Verlag 1978
I. Wegener	Skript Effiziente Algorithmen
Kapitel 5.3	Uni Dortmund 1991

Version 1.0 vom 05.02.01

Michael Daumen mdaumen@tabu.uni-bonn.de