Minimale Spannbäume

Evangelia Tsiouprou

LMU
Proseminar Computer Unplugged
WS 2004/05

Minimale Spannbäume

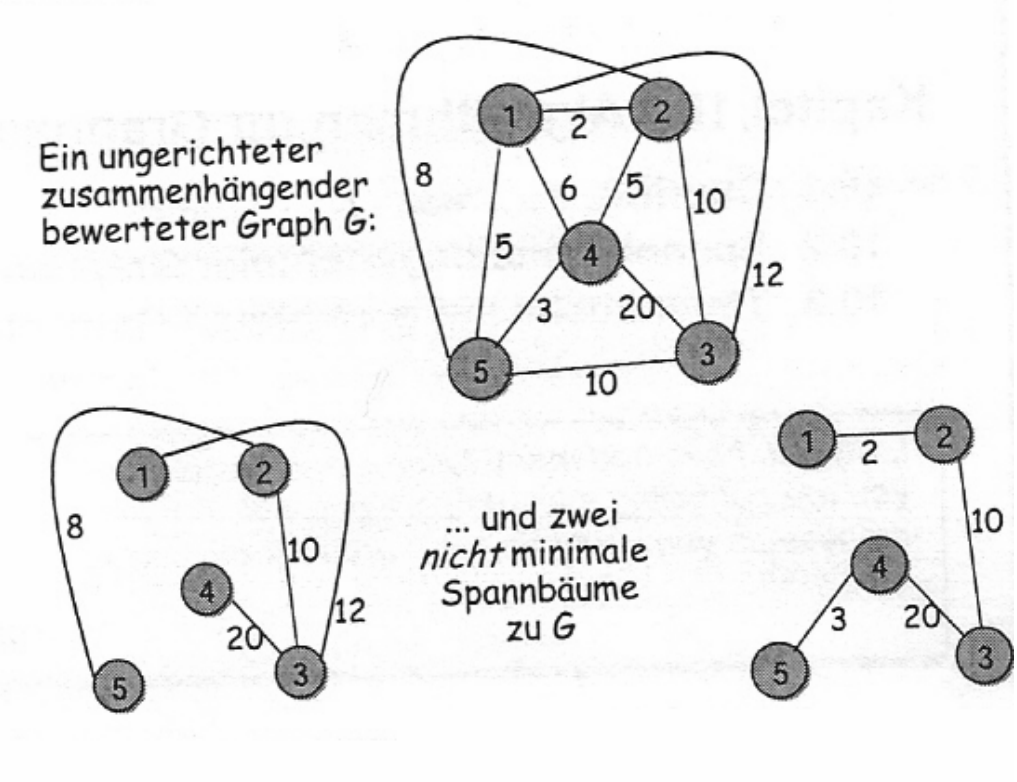
1 Motivation

Um Bäume auf Verbindungen zwischen einzelnen Knoten zu untersuchen, wird auf die sogenannten Spannbäume zurückgegriffen. Um die Verbindungen möglichst kostengünstig zu halten, werden minimale Spannbäume untersucht.

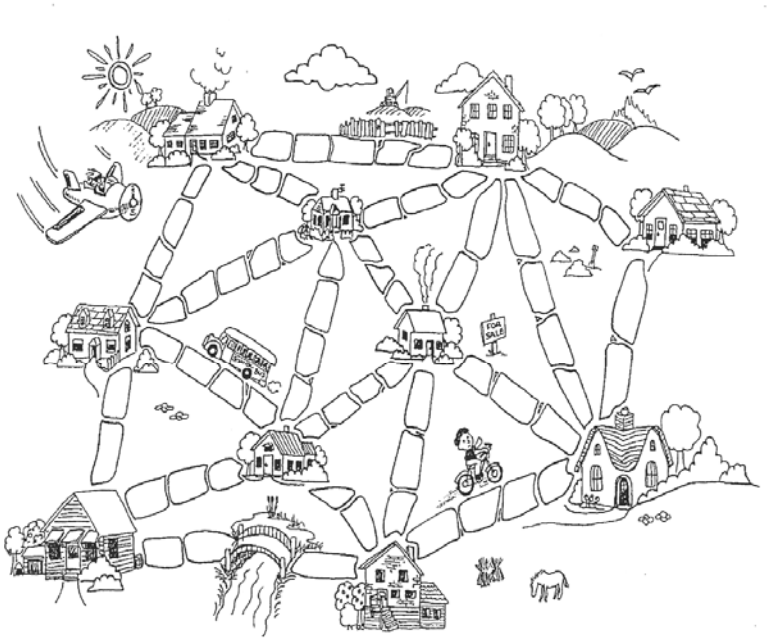
1.1 Spannbaum

Sei G = (N, E) mit N = Knotenmenge und E = Menge ungerichteter Kanten ein zusammenhängender ungerichteter Graph. Ein ebenfalls zusammenhängender Teilgraph T von G, der dieselbe Knotenmenge N besitzt und Baum ist, heißt *Spannbaum* von G.

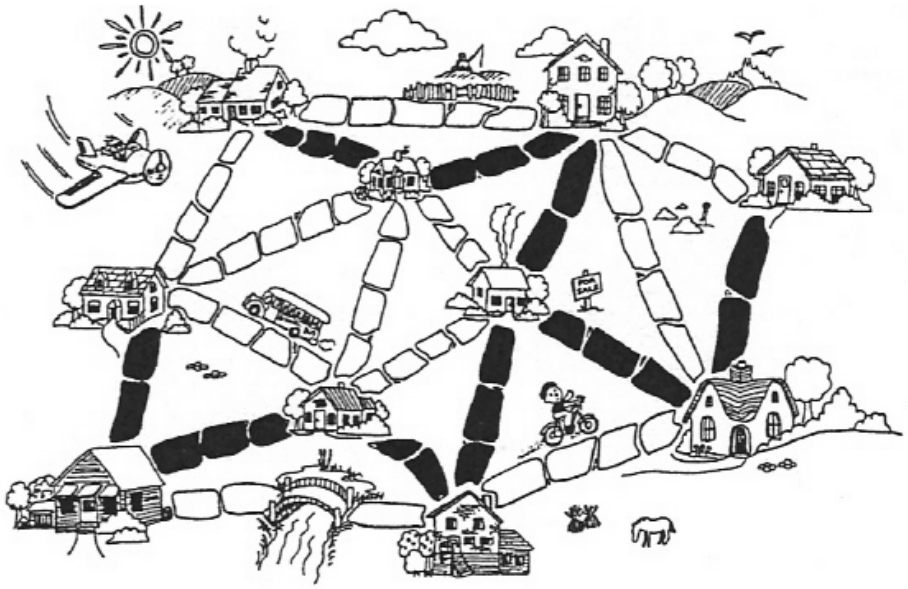
Beispiel:

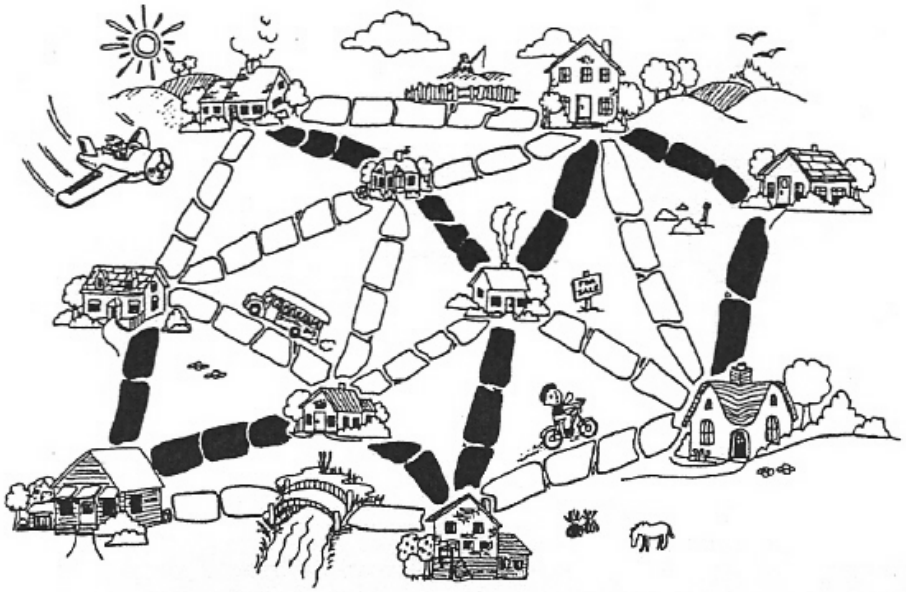


Ein Beispiel aus dem Buch Computer Science Unplugged von Tim Bell, Ian H. Witten und Mike Fellows. Wir wollen alle Häuser des Dorfes möglich schnell besuchen. Eie minimaler Spannbaum des Dorfes wird gesucht.



Und nun Vorschläge: Zwei verschiedene Spannbäume des Dorfes im oben gezeichneten Bild. Beide Lösungen sind richtig.





2 Algorithmen zur Berechnung eines MST:

2.1 Der Algorithmus von Kruskal

Dieser Algorithmus wurde 1930 von JARNIK entworfen und von KRUSKAL im Jahre 1956 wiederaufgegriffen.Der Algorithmus von Kruskal

liefert zu einem bewerteten ungerichteten Graphen G einen minimalen aufspannenden Baum.
Er arbeitet nach einem sogenannten gierigen (englisch: greedy) Verfahren. Das heißt:
Entscheidungen die während des Verfahren getroffen werden, sind nicht reversibel, sie können zu keinem späteren Zeitpunkt rückgängig gemacht werden.

Für das Greedy-Prinzip gilt:
"Lokales Optimum führt zu globalem Optimum."

Es wird also stets die lokal günstigste Möglichkeit gewählt.
Das Greedy-Prinzip ist eines der wichtigsten algorithmischen Prinzipien. Problembeispiele, welche mittels dieser "Nimm-was-du-kriegen-kannst"-Strategie gelöst werden können, wären:

- Finde die beste Reihenfolge, um in kürzester Zeit eine Menge von Aufträgen abzuarbeiten.
- Finde die beste Reihenfolge, eine Menge von Aufträgen mit maximalem Gewinn zu
bearbeiten, so dass jeder Auftrag noch vor Schlusstermin erledigt ist.
- Finde die kürzesten Abstände und die zugehörigen kürzesten Pfade von einem
gegebenen Knoten eines Graphen zu allen anderen Knoten
(Single-source shortest paths problem)
Die Greedy-Strategie muss aber nicht immer zu einer optimalen Lösung führen!
Es gibt auch Probleme, bei deren Lösung er versagt, zum Beispiel:
- Gib einem Kunden Wechselgeld heraus und verwende dazu möglichst wenig Münzen.
Der Kruskal-Algorithmus arbeitet mit einer sortierten Kantenliste. Diese setzt sich aus allen Kanten zusammen, welche im Ausgangsgraph enthalten sind.
Jede Kante dieser Liste wird betrachtet und wird anschließend entweder verworfen oder für die Konstruktion des MST ausgewählt. Der minimale Spannbaum wird so sequentiell aus Teilbäumen aufgebaut.

Verfahren:

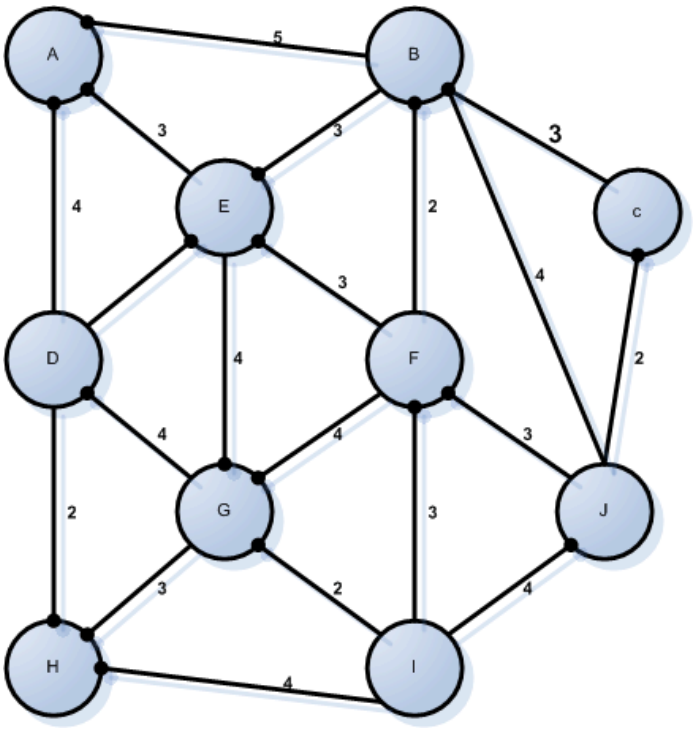
(1) START mit leerer Kantenmenge;
Jeder Knoten steht als eigene Komponente, er ist ein gewählter Baum im Graph
(2) Alle Kanten des Ausgangsgraphen nach Gewichtung sortieren
(3) Kante mit geringster Gewichtung nehmen (Greedy-Prinzip!)
(4) Untersuchen:
Sind die Knoten, die durch diese Kante verbunden werden, in unterschiedlichen
Komponenten?
(oder anders formuliert:)
Entsteht kein Zyklus, wenn diese Kante aufgenommen wird?
Wenn JA: Kante aufnehmen und Komponenten vereinigen,

Kante aus sortierter Kantenliste löschen
Wenn NEIN: Kante verwerfen und aus sortierter Kantenliste löschen
(5) Solange sortierte Kantenliste nicht leer ist, weiter mit (3)
(6) ENDE

Wenn alle Kanten verarbeitet wurden, erhält man aus allen Knoten und allen Zugefügten Kanten einen minimalen Spannbaum des Graphen. Dieser muß allerdings nicht eindeutig sein! Ein Graph kann, wie oben schon einmal bemerkt, durchaus mehrere MST besitzen. Es ist bei der Ausführung des Algorithmus erforderlich, das man zu jedem Zeitpunkt von jedem Knoten weiß, zu welcher
Zusammenhangskomponente er gehört. Kritisch für die Gesamtlaufzeit des Algorithmus ist die Sortierung aller Kanten nach den Gewichten. Die Sortierung der Kantenliste kann durch einen geeigneten Sortieralgorithmus wie Mergesort durchgeführt werden.

2.1.1 Das Dorfbeispiel für den Algorithmus von Kruskal

Gegeben sei der Graph des Dorfbeispiels mit den Häusern als Knoten und die Menge der Steine als nummerierte Kanten:



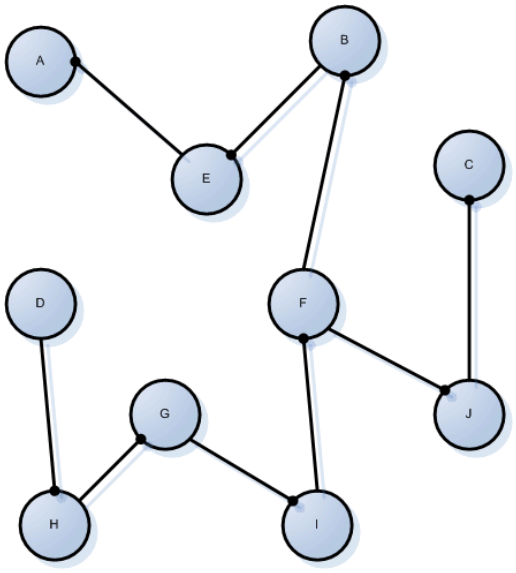
Zu diesem Graph ergibt sich folgende sortierte Kantenliste:

Kante	Gewichtung
(C,,J)	2
(B,F)	2
(G,I)	2
(D,H)	2
(C,B)	3
(J,F)	3
(F,I)	3
(G,H)	3
(F,E)	3
(B,E)	3
(E,A)	3
(B,J)	4
(J,I)	4
(I,H)	4
(D,G)	4
(G,E)	4
(G,F)	4
(A,D)	4
(D,E)	5
(A,B)	5

Nun entfernt man alle Kanten aus dem Graphen und wendet den Kruskal-Algorithmus an. So werden nacheinander folgende Kanten eingefügt:

(C,J) , (J,F) , (F,B) , (B,E),(E,A),(F,I),(I,G),(G,H),(H,D)

Nun sind alle Knoten des Graphen miteinander verbunden. Die verbliebenen Kanten der Liste werden verworfen. Es ist ein MST entstanden.



2.2 Der Algorithmus von Prim

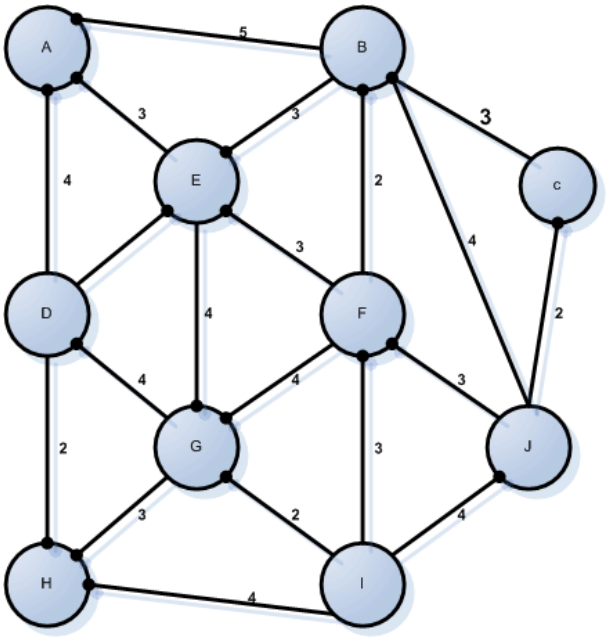
Dieser Algorithmus geht im Wesentlichen auf JARNIK zurück und wurde von DIJKSTRA und PRIM wiederentdeckt. Er ist auch unter dem Namen Prim-Dijkstra-Algorithmus bekannt.
Der Algorithmus von Prim liefert, ebenso wie der Kruskal-Algorithmus,durch ein gieriges Verfahren einen minimalen aufspannenden Baum eines bewerteten ungerichteten Graphen. Allerdings läuft er schneller ab, ist also im Vergleich effektiver. Es wird von einem beliebigen Startknoten s ausgegangen. Zur Laufzeit bilden die gewählten Kanten zu jedem Zeitpunkt einen einzigen Baum. Bei insgesamt n Knoten wird dann folgender Schritt (n-1)mal ausgeführt:
Man wählt eine Kante mit minimaler Länge, die genau einen Endknoten zum gewählten
Baum gehört. Zuerst besteht der Baum nur aus dem Startknoten s, später dann besteht der
Baum aus den gewählten Kanten mit deren inzidenten Knoten.

Verfahren:

(1) START mit leerer Knotenmenge U
(2) Beliebigen Knoten wählen und in Knotenmenge U aufnehmen
(3) (| V | - 1)mal wiederholen:
- Suche Knoten v □ V \ U mit dem geringsten Abstand zu u □U
(Greedy-Prinzip!)
- Nimm Kante auf
- Nimm Knoten v in Knotenmenge U auf
Das Verfahren ähnelt dem Verfahren von Dijkstra zur Berechnung kürzester Wege.

2.2.1 Das Dorfbeispiel für den Algorithmus von Prim

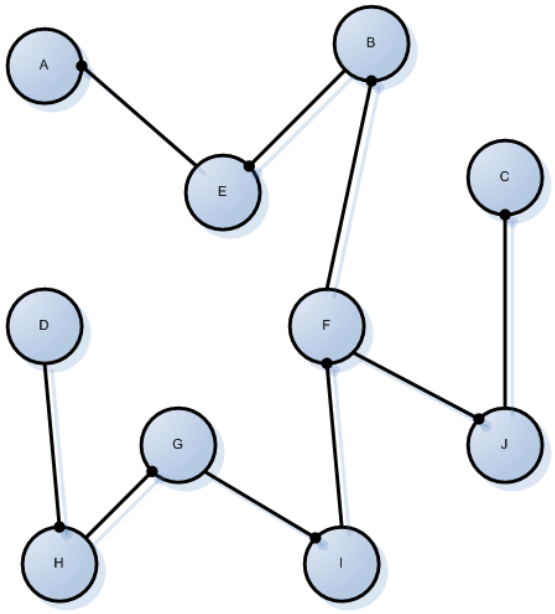
Gegeben sei der Graph des Dorfbeispiels mit den Häusern als Knoten und die Menge der Steine als nummerierte Kanten:



Als Startknoten wird Knoten C gewählt.
Nun wird mit der Abarbeitung des Algorithmus begonnen.

Schritt Nr.	**Knotenmenge U**	**V \ U**	**zu wählende Kante**	**Kantengewicht**
0	{ C }	{ A,B,D,E ,F,G,H,I,J}	(C,J)	2
1	{ C,J }	{ A,B,D,E, F,G,H,I}	(J,F)	**3**
2	{ C,J,F }	{A,B,D,E,G,H,I}	**(F,B)**	**2**
3	**{ C,J,F,B}**	**{A,D,E,G,H,I}**	**(B,E)**	**3**
4	**{C,J,F,B,E}**	**{A,D,G,H,I}**	**(E,A)**	**3**
5	**{C,J,F,B,E,A}**	**{D,G,H,I}**	**(F,I)**	**3**
6	**{C,J,F,B,E,A,I}**	**{D,G,H,}**	**(I,G)**	**2**
7	**{C,J,F,B,E,A,I,}**	**{D,H,}**	**(G,H)**	**3**
8	**{C,J,F,B,E,A,I,H}**	**{D}**	**(H,D)**	**2**

Es ist ein MST entstanden.



3 Travelling Salesman Problem

Das Travelling Salesman Problem (TSP) oder „Problem des Handlungsreisenden“ besteht darin, daß ein Handlungsreisender eine Rundreise durch n Städte unternehmen und dabei einen möglichst kurzen Weg zurücklegen soll.
Die Entfernungen zwischen den einzelnen Städten sind bekannt. Gefragt ist also nach der Reihenfolge, in der die Städte besucht werden müssen.
Das TSP fällt in die Kategorie der NP-vollständigen Probleme (non-deterministic polynominal). Unter diesem Punkt werden all die Probleme zusammengefaßt, für die eine optimale Lösung existiert, doch diese ist immer nur in ex Das Travelling Salesman Problem (TSP) oder „Problem des Handlungsreisenden“ besteht darin, daß ein Handlungsreisender eine Rundreise durch n Städte unternehmen und dabei einen möglichst kurzen Weg zurücklegen soll.
Mathematiker und Informatiker suchen einen Algorithmus, der weniger als exponentielle Zeit benötigt um das TSP zu lösen. Solange dieser nicht gefunden ist, stellt sich die Frage nach approximativen Verfahren, die zwar nicht unbedingt die beste Lösung finden, aber dieser sehr nahe kommen.

3.1 Metrisches TSP

Wenn die symmetrische Gewichtsmatrix W=(ωij) die Dreiecksungleichung $\omega_{ik} <= \omega_{ij} + \omega_{jk}$ für alle i,j,k = 1,...,n erfüllt, so spricht man von einem metrischen TSP oder auch ΔTSP.
Bei Matrizen, die Abstände in der Ebene oder in einem Graphen angeben oder zu einem metrischen Raum gehören, ist das immer der Fall.

3.2 Asymmetrische TSP

Neben dem metrischen TSP existiert auch ein asymmetrischer TSP (ATSP), Dieser beinhaltet den gewöhnliche TSP als Spezialfall und zeichnet sich dadurch aus, daß die Gewichtsmatrix auf der Hauptdiagonalen kein Null-Einträge besitzt. Dieses sei aber nur am Rande erwähnt und soll nicht weiter verfolgt werden.

3.3 Erläuterung Beispiel

Bei diesem Beispiel handelt es sich um die Bestimmung einer optimalen Tour zwischen den Städten Paris (Pa), London (Lo), New York (NY), Los Angeles (LA), Peking (Pe), Tokyo (To), Osaka (Os) und Brasilia (Br). Die Entfernungen sind in der Gewichtsmatrix wiedergegeben (in 100 Meilen) :

	Lo	LA	NY	Pa	Pe	To	Br	Os
Lo	0	69	43	2	66	81	58	80
LA	69	0	27	71	66	51	64	54
NY	43	27	0	44	91	77	53	81
Pa	2	71	44	0	65	79	57	78
Pe	66	66	91	65	0	15	105	13
To	81	51	77	79	15	0	109	3
Br	58	64	53	57	105	109	0	105
Os	80	54	81	78	13	3	105	0

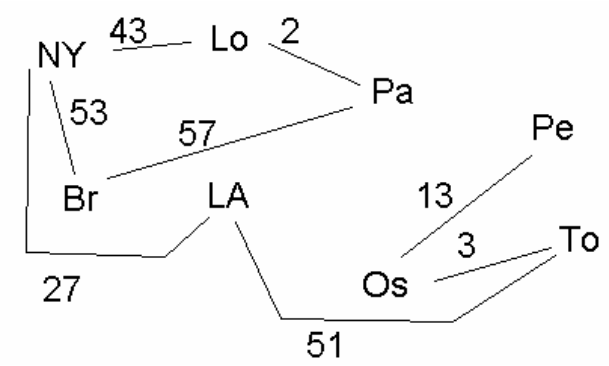
3.4 MST-Relaxation

Hierbei handelt es sich um die Bestimmung des minimalen erzeugenden Baumes für K_n unter Beachtung seiner Kantengewichte. Dazu existieren diverse Algorithmen wie Prim oder Kruskal, die hier problemlos angewandt werden können.

Natürlich ist eine Tour kein Baum, aber durch entfernen einer beliebigen Kante aus der Tour erhält man einen Hamiltonischen Weg. Unter einen Hamilitonischen Weg versteht man einen einfachen Weg , der jeden Punkt von K_n enthält.
Trotzdem gilt ω(MST) <= ω(TSP). Den Wert ω von MST läßt sich, wie schon oben erwähnt, mittels Prim- oder Kruskalalgorithmus bestimmen.

3.5 Erläuterung am Beispiel

Für das Beispiel entsteht ein minimaler Spannbaum mit Brasilia als Punkt 1, da hier die Summe der zwei kleinsten Kantengewichte mit 110 am größten ist. Für alle restlichen Städte wird der minimale Spannbaum erzeugt und die zwei billigsten Kanten von Brasilia hinzugefügt. Nach Ausführung des Algorithmus entsteht ein Spannbaum der Größe 249.



Es existieren noch weitere Relaxationen wie zum Beispiel Assignment- und LP-Relaxation. Auf diese soll aber an dieser Stelle nicht weiter eingegangen werden.

Mit Hilfe der Subgradientenoptimierung können die Schranken noch weiter verfeinert werden. Dieses Verfahren ist aber sehr aufwendig und ist für die weitere Ausarbeitung nicht weiter relevant.

4 Christofides-Algorithmus

Der Christofides-Algorithmus ist ein ½-approximativer-Algorithmus und weist eine Komplexität von $O(n^3)$ beim metrischen TSP auf. Er ist damit der bisher beste bekannte Approximativalgorithmus für das ΔTSP.

Im Folgenden werden noch 2 grundlegende Definition erläutert. Anschließend soll die Funktionsweise des Christofides-Algorithmus gezeigt werden sowie an dem Beispiel näher erklärt werden.

4.1 Definition erzeugender Eulersche Multigraph

Ein **Multigraph** ist ein Graph, der Punktpaare enthält, welche durch mehrere Kanten verbunden sind.
Ein **Eulerscher Kreis** ist ein Kreis, der jeder Kante eines **Multigraphen** G genau einmal enthält.
Ein Multigraph heißt **Eulersch**, wenn er einen **Eulerschen Kreis** besitzt.

Bei einem gegebenen metrischen TSP auf dem K_n und dem erzeugenden Eulerschen Multigraphen G kann man eine Tour π mit $\omega(\pi) <= \omega(E)$ (E ist Kantenmenge von G) und der Komplexität O(|E|) konstruieren [Jung94, 531].

4.2 Definition Perfektes Matching

Unter **Matching** versteht man eine Menge von Kanten mit paarweisen verschiedenen Endknoten.
Ein Matching heißt **perfekt**, falls es alle Knoten des Graphen überdeckt, d. h. falls jeder Knoten gematcht ist. Daraus resultiert, daß nur Graphen gerader Ordnung ein perfektes Matching besitzen können aber nicht müssen!

4.3 Funktionsweise Christofides-Algorithmus

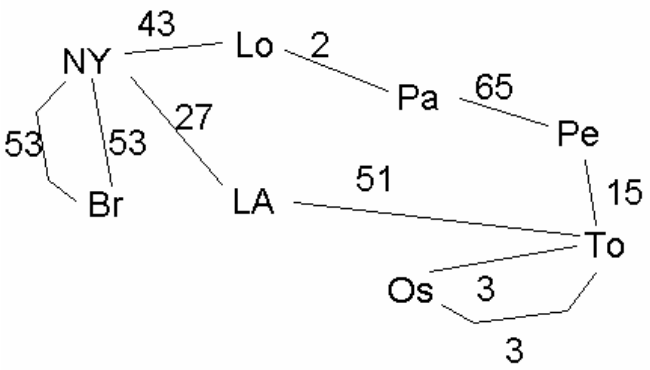
Gegeben sei ein ΔTSP auf dem K_n mit der Gewichtsmatrix $W=(\omega_{ij})$.

1.) Man bestimme den minimalen Spannbaum T des K_n bezüglich W.
2.) Es sei X die Menge derjenigen Punkte, die in T einen ungeraden Grad besitzen.
3.) Sei der vollständige Graph auf X mit der Einschränkung von W als Gewichtsfunktion. Man bestimme ein perfektes Matching K minimalen Gewichts in H.
4.) G sei der Multigraph, der aus T durch Hinzufügen des Kanten von K entsteht. Man bestimme einen Eulerschen Kreis C von G.
5.) Man wähle eine in C enthaltene Tour.

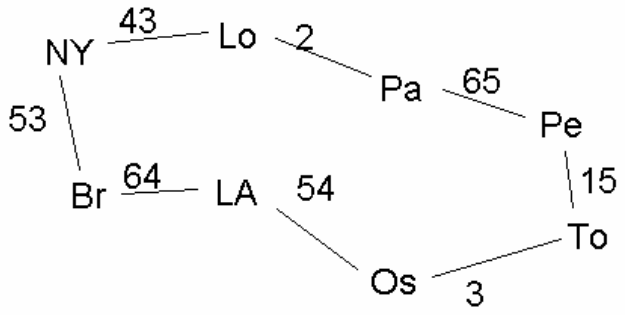
4.4 Erläuterung am Beispiel

Man greift auf den minimal erzeugten Baum T zurück.
Die Menge aller Punkte mit ungeradem Grad ist hier X = {Br,NY,Pa,Pe,To,Os}. Man muß also ein perfektes Matching minimalen Gewichts bezüglich der Gewichtsmatrix aller Punkte von X suchen. Man erhält K = {BrNy,PaPe,ToOs} mit ω(K) = 121.

Die Kanten von K werden dem minimalen Spannbaum T hinzugefügt und man erhält einen Eulerschen Multigraphen vom Gewicht 315 mit dem Eulerschen Kreis (Lo,Pa,Pe,To,Os,To,LA,NY,Br,NY,Lo).

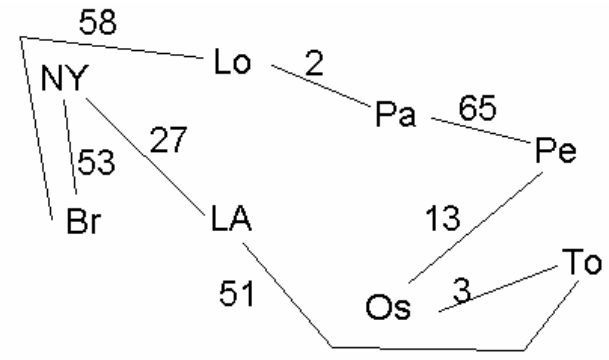


Die darin enthaltene Tour (Lo,Pa,Pe,To,Os,LA,Br,NY,Lo) ha t das Gewicht



299.

Mit Hilfe des Computers wurde die optimale Tour (Lo,Pa,Pe,Os,To,La,NY,Br,Lo) mit 272 errechnet (siehe unten). Die Tour des Christofides liegt somit nur ca. 10% über der optimalen Tour!



5 Literaturverzeichnis

- Tim Bell, Ian H. Witten und Mike Fellows; Das Buch Computer Science Unplugged
- THIENEL (1997/1998); Der Handlungsreisende und seine Verwandte; Skript vom Informatiklehrstuhl der Universität Köln