

Kürzeste Wege, Spanner und Netzwerkdesign

Bassam Alrifaae

TU Ilmenau

18.06.09

Gliederung

1. Einführung
 - Motivation
 - Vereinbarung
2. Berechnung kürzester Wege in Graphen
3. Leichte angenäherte Kürzeste-Wege-Bäume
 - Die Präordernummerierung von Bäumen
 - Die Berechnung eines LAST
4. Spanner
 - Analyse des Stretch-Faktors
 - Analyse der Kantenzahl des Spanners
 - Analyse des Gewichts des Spanners
5. Modellierung eines Netzwerkdesign-Problems
6. Design von Access-Netzen
 - Ein Approximationsalgorithmus für das Design von Access-Netzen
7. Allgemeines Netzwerkdesign
 - Ein Approximationsalgorithmus für Allgemeines Netzwerkdesign

- Ziel des Netzwerkdesigns: kostengünstiges Netz zu entwerfen
 - Topologie und Routing festlegen → Verkehrsanforderungen gut erfüllen
- Abhängig von der Netzwerk-Architektur, den zu verwendenden Protokollen, Kundenwünsche usw.
 - → kein einzelnes Problem, das „das Netzwerkdesign-Problem“ heißt
 - sondern eine Vielzahl verwandter Probleme, die alle unter dem Begriff Netzwerkdesign zusammengefasst werden
- Hier behandelt: Zwei konkrete Probleme
- Algorithmen zur Berechnung von Kürzeste-Wege-Bäumen, angenäherten Kürzeste-Wege-Bäumen und Spanners als Unterroutrinen sind nötig

- Sei ein ungerichteter Graph $G = (V, E)$ mit Kantengewichten (Kosten) $c : E \rightarrow \mathbb{R}^+$ gegeben
- Im Hinblick auf Anwendungen beim Netzwerkdesign seien:
 - V die Menge der Knoten, die wir vernetzen wollen
 - die Kanten in E alle Links, die wir potentiell verwenden könnten
 - $c(e)$ (das Gewicht einer Kante e) die Kosten, die die Verwendung des Links e mit sich bringen würde
 - ein Knoten $s \in V$ (source, Wurzel) ausgezeichnet, mit dem alle anderen Knoten verbunden werden sollen

Berechnung kürzester Wege in Graphen

- Geg. $G = (V, E)$ mit $|V| = n$, $|E| = m$ und $c : E \rightarrow \mathbb{R}^+$
- Ziel: Für einen Startknoten $s \in V$ berechne die kürzesten Wege (Pfade) zu allen anderen Knoten in V
→ Effiziente Lösung: Dijkstra-Algorithmus, Laufzeit $O(n \log n + m)$

- Spannbäume eines Graphen, die die Eigenschaften von *minimalen Spannbäumen* und von *kürzeste-Wege-Bäumen* in gewissem Sinne vereinen
- Ein minimaler Spannbaum in G ist ein *Spannbaum* $T = (V, E')$ von G , so dass die Summe der Kantengewichte $\sum_{e \in E'} c(e)$ minimal ist und kann z.B. mit dem Algorithmus von Prim effizient in Zeit $O(n \log n + m)$ berechnet werden
- Bezeichne mit $T_M(G)$ einen minimalen Spannbaum von G , mit $c(T_M(G))$ sein Gewicht und mit $d_T(s, v)$ die Länge des Pfades von s nach v in T
- Ein *Kürzester-Wege-Baum* für G und $s \in V$ ist ein Spannbaum von G , so dass für jeden Knoten $v \in V$ die Länge des Pfades (Summe der Gewichte aller Kanten auf dem Pfad) von s nach v in dem Baum gleich der Länge eines kürzesten Pfades von s nach v in G und kann mit dem Algorithmus von Dijkstra effizient in Zeit $O(n \log n + m)$ berechnet werden
- Bezeichne mit $d_G(s, v)$ die Länge des kürzesten Pfades von s nach v in G

Definition:

▪ Für gegebene Parameter $\alpha \geq 1$ und $\beta \geq 1$ nennen wir einen Spannbaum T von G einen (α, β) -leichten angenäherten Kürzeste-Wege-Baum (kurz (α, β) -LAST, wegen englisch light approximate shortest-paths tree), wenn:

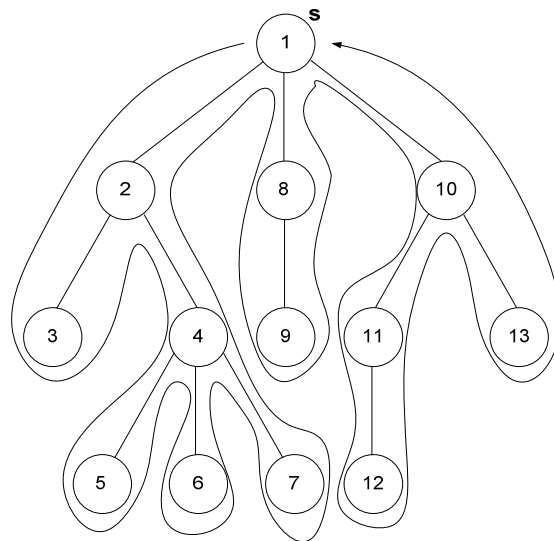
1. $\forall v \in V : d_T(s, v) \leq \alpha d_G(s, v)$

2. $c(T) \leq \beta c(T_M(G))$, wobei $c(T)$ das Gesamtgewicht aller Kanten von T bezeichnet

➤ Ziel: Finde einen Algorithmus, der für möglichst kleine Werte von α und β *effizient* einen (α, β) -LAST berechnet

Definition:

Eine Präordernummerierung eines Baumes ist eine bijektive Zuordnung der Zahlen $1, 2, \dots, n$ zu den Knoten des Baumes in der Reihenfolge, wie die Knoten in einem bei der Wurzel beginnenden Präorderdurchlauf besucht werden. Ein Präorderdurchlauf ist ein Durchlauf des Baumes, bei dem nach dem Besuch eines Knotens nacheinander Präorderdurchläufe in den an seinen Kindern gewurzelten Teilbäumen durchgeführt werden.



Algorithmus: Präordernummerierung (Rekursiv)

Eingabe: Baum $T = (V, E)$ mit Wurzel $s \in V$

Ausgabe: Präordernummern $num[v]$ für alle $v \in V$

procedure preorder(node v)

begin

$num[v] := i, i++;$

for alle Kinder w von v **do**

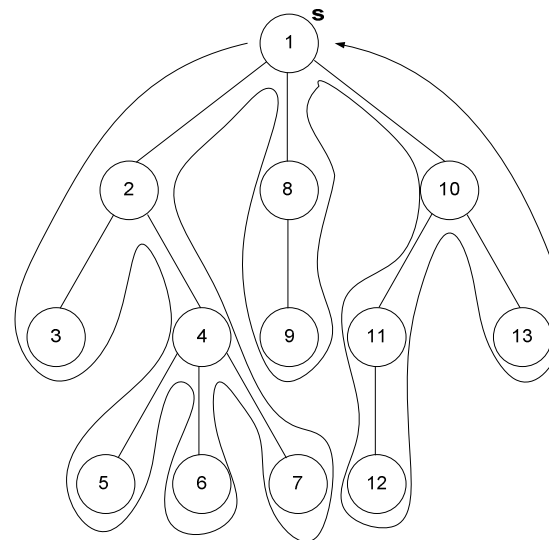
 preorder(w);

od

end

$i := 1;$

preorder(s);



Lemma1:

Sei T ein Spannbaum von G . Betrachte s als die Wurzel von T . Wenn z_0, z_1, \dots, z_k beliebige $k+1$ verschiedene Knoten von T in Reihenfolge aufsteigender Präordernummern sind, dann gilt:

$$\sum_{i=1}^k d_T(z_{i-1}, z_i) \leq 2c(T)$$

Beweis:

Betrachte den (nicht einfachen) Weg w , den ein Präorderdurchlauf durch T zurücklegt. Dieser Weg beginnt und endet bei der Wurzel s und durchläuft jede Kante des Baums genau zweimal, einmal abwärts und einmal aufwärts. Die Summe der Kantengewichte auf dem Weg w ist $2c(T)$.

Für jedes i , $1 \leq i \leq k$, enthält der Weg w einen Teilpfad vom ersten Vorkommen von z_{i-1} bis zum ersten Vorkommen von z_i . Die Länge dieses Teilpfads von w ist mindestens $d_T(z_{i-1}, z_i)$. Ausserdem sind alle k solchen Teilpfade disjunkte Teile von w , da die Knoten z_i in Reihenfolge aufsteigender Präordernummern gegeben sind.

□

Algorithmus: (α, β) -LAST

Eingabe: zusammenhängender ungerichteter Graph $G = (V, E)$, ausgezeichneter Knotens $s \in V$,
Kantengewichte $c : E \rightarrow \mathbb{R}^+$, Werte (α, β) mit $\alpha > 1$ und $\beta = 1 + \frac{2}{\alpha - 1}$

Ausgabe: (α, β) -LAST T

1. berechne minimalen Spannbaum T_M von G ;
2. berechne kürzeste Pfade von s zu allen anderen Knoten in G ;
3. berechne Präordernummerierung von T_M mit s als Wurzel;
4. $H := T_M$
for alle Knoten $v \in V$ in Reihenfolge aufsteigender Präordernummern **do**
 berechne kürzesten Pfad p von s nach v in H ;
 if $c(p) > \alpha d_G(s, v)$ then
 füge einen kürzesten Pfad von s nach v in G in den Graphen H ein;
 fi
od
5. berechne in H einen Kürzeste-Wege-Baum T mit Startknoten s ;
6. **return** T ;

Satz:

Der Algorithmus berechnet in polynomieller Zeit einen (α, β) -LAST.

Beweis:

a) $\forall v \in V : d_T(s, v) \leq \alpha d_G(s, v)$: H enthält einen Pfad von s nach v mit Länge höchstens $\alpha d_G(s, v)$, und wir berechnen am Ende in H einen Kürzeste-Wege-Baum mit Startknoten s .

b) $c(T) \leq \beta c(T_M(G))$: wir zeigen, dass sogar $c(H) \leq \beta c(T_M(G))$ gilt.

Seien z_1, z_2, \dots, z_k diejenigen Knoten, für die die if-Bedingung erfüllt war. Wähle $z_0 = s$.

Nach Lemma 1 erhalten wir:
$$\sum_{i=1}^k d_T(z_{i-1}, z_i) \leq 2c(T) \quad (1)$$

Als der Algorithmus in Schritt 4 den Knoten z_i bearbeitete, enthielt H auf jeden Fall den Pfad von s nach z_i , der aus dem kürzesten Pfad von s nach z_{i-1} in G und dem Pfad von z_{i-1} nach z_i in T_M besteht. Dieser Pfad hat Länge $d_G(s, z_{i-1}) + d_{T_M}(z_{i-1}, z_i)$. Da der kürzeste Pfad von s nach z_i in H zu diesem Zeitpunkt länger als $\alpha d_G(s, z_i)$ war, muss also gelten: $d_G(s, z_{i-1}) + d_{T_M}(z_{i-1}, z_i) > \alpha d_G(s, z_i)$

→


Umformen zu $\alpha d_G(s, z_i) - d_G(s, z_{i-1}) < d_{T_M}(z_{i-1}, z_i)$

Wenn wir diese Ungleichungen für $i = 1, 2, \dots, k$ aufsummieren, erhalten wir (unter Berücksichtigung von $d_G(s, z_0) = 0$):

$$\begin{array}{rcl}
 \alpha d_G(s, z_1) - d_G(s, z_0) & < & d_{T_M}(z_0, z_1) \\
 \alpha d_G(s, z_2) - d_G(s, z_1) & < & d_{T_M}(z_1, z_2) \\
 & \vdots & \\
 \alpha d_G(s, z_k) - d_G(s, z_{k-1}) & < & d_{T_M}(z_{k-1}, z_k) \\
 \hline
 \alpha d_G(s, z_k) + \sum_{i=1}^{k-1} (\alpha - 1) d_G(s, z_i) & < & \sum_{i=1}^k d_{T_M}(z_{i-1}, z_i)
 \end{array}$$

Mit der Ungleichung (1) erhalten wir $\sum_{i=1}^k d_G(s, z_i) < \frac{2}{\alpha - 1} c(T_M)$

Es gilt nach Konstruktion des Algorithmus: $c(H) \leq c(T_M) + \sum_{i=1}^k d_G(s, z_i)$


 $c(H) \leq \left(1 + \frac{2}{\alpha - 1}\right) c(T_M)$

□

- Ein (α, β) -LAST enthält für *einen bestimmten Knoten* Wege zu allen anderen Knoten, die nicht viel länger als die kürzesten Wege im Ursprungsgraphen sind
- Nun wollen wir einen Teilgraphen finden, in dem die Wege für *alle Paare von Knoten* nicht viel länger als im Ursprungsgraphen sind

Definition:

- Sei $G = (V, E)$ ein ungerichteter zusammenhängender Graph mit Kantengewichten

$$c : E \rightarrow \mathbb{R}^+$$

- Bezeichne mit $d_G(u, v)$ die Länge eines kürzesten Pfades von u nach v in G
- Sei $H = (V, E')$ ein zusammenhängender Teilgraph von G , der alle Knoten von G enthält (spannender Teilgraph)
- Bezeichne mit $d_H(u, v)$ die Länge eines kürzesten Pfades von u nach v in H
- Definiere: $Stretch(H) = \max_{u, v \in V} \frac{d_H(u, v)}{d_G(u, v)}$ Faktor, um den der kürzeste Pfad zwischen zwei Knoten in H höchstens länger sein kann als in G
- Wenn $Stretch(H) \leq t$ für ein $t \geq 1$ gilt, so nennen wir H einen t -Spanner von G

Algorithmus: Greedy-Spanner

Eingabe: Zusammenhängender Graph $G = (V, E)$ mit Kantengewichten $c : E \rightarrow \mathbb{R}^+$

Parameter $t > 1$

Ausgabe: Zusammenhängender Teilgraph $H = (V, E')$ von G

sortiere die Kanten in $E = \{e_1, e_2, \dots, e_m\}$ so, dass $c(e_1) \leq c(e_2) \leq \dots \leq c(e_m)$;

$E' := \Phi$;

$H := (V, E')$;

for $i = 1$ to m **do**

 // sei $e_i = \{u, v\}$

if $d_H(u, v) > t \cdot c(\{u, v\})$ **then**

$E' := E' \cup \{e_i\}$;

$H := (V, E')$;

fi

od;

return H ;

Bemerkung: Falls in H noch gar kein Pfad von u nach v enthalten ist, so ist $d_H(u, v) = \infty$ und die Kante $\{u, v\}$ wird auf jeden Fall in H eingefügt.

Spanner

- Bezeichne mit $c(H)$ die Summe der Gewichte aller Kanten in H
- Bezeichne mit $MST(G)$ das Gewicht eines minimalen Spannbaums in G

Satz:

Der Algorithmus Greedy-Spanner liefert für einen Graphen $G = (V, E)$ mit n Knoten und Kantengewichten $c : E \rightarrow \mathbb{R}^+$ einen Teilgraphen H , für den gilt:

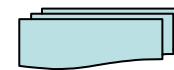
(a) H ist ein t -Spanner von G .

(b) H enthält höchstens $n \cdot \left\lceil n^{\frac{2}{t-1}} \right\rceil$ Kanten.

(c) Für jedes δ mit $0 < \delta < \min\{t-1, 1\}$ gilt: $c(H) \leq \left(5 + \frac{32t}{\delta^2}\right) \cdot n^{\frac{2+\delta}{t-1-\delta}} \cdot MST(G)$

Der Satz liefert obere Schranken sowohl für die Anzahl der Kanten von H als auch für das Gesamtgewicht aller Kanten von H .

Den **Beweis** teilen wir in eine Reihe von Lemmas auf!!!



Lemma:

Der vom Algorithmus Greedy-Spanner berechnete Teilgraph $H = (V, E')$ ist ein t -Spanner von $G = (V, E)$.

Beweis:

Betrachte eine Kante $e = \{u, v\}$, die in G enthalten ist, aber nicht in H . Als der Algorithmus die Kante e bearbeitet hat, muss H bereits einen Pfad von u nach v der Länge höchstens $t \cdot c(e)$ enthalten haben. Also enthält H für jede Kante $e = \{u, v\} \in E$ einen Pfad von u nach v der Länge höchstens $t \cdot c(e)$.

Seien x und y zwei beliebige Knoten in V und sei p ein kürzester Pfad von x nach y in G . Wir können jede Kante e des Pfades p durch einen Pfad der Länge höchstens $t \cdot c(e)$ in H ersetzen und erhalten so einen Pfad von x nach y der Länge höchstens $t \cdot d_G(x, y)$ in H .

→ $Stretch(H) \leq t$.

□

Lemma:

Sei H ein ungerichteter Graph mit n Knoten und m Kanten, in dem jeder Kreis aus mindestens g Kanten besteht. Dann gilt

$$m \leq n \cdot \left\lceil n^{\frac{2}{g-2}} \right\rceil \leq 2 \cdot n^{1+\frac{2}{g-2}}$$

Beweis:

- Falls $m \leq 2n$ ist die Behauptung offensichtlich richtig.
- Falls $m > 2n$: Sei $k = \left\lfloor \frac{m}{n} \right\rfloor + 1$. Es gilt $k \geq 3$. Solange H einen Knoten mit Grad kleiner als k

enthält, entfernen wir diesen Knoten und alle seine inzidenten Kanten aus H . Dadurch können nicht alle Knoten entfernt werden, weil wir beim Entfernen der ersten $n-1$ Knoten höchstens $(n-1)(k-1) < m$ Kanten entfernen und dann keine Kante mehr übrig sein kann, ein Widerspruch. Also bleibt ein Teilgraph H' von H übrig, in dem alle Knoten Grad mindestens k haben.

- Falls g ungerade $g = 2d + 1$

Sei x ein beliebiger Knoten in H' und betrachten wir alle Knoten, die von x aus über höchstens d Kanten erreicht werden können. Diese Knoten müssen einen Baum bilden, da alle Kreise in H und damit auch in H' aus mindestens $2d + 1$ Kanten bestehen. Keine zwei Knoten in diesem Baum können identisch sein, da H' sonst einen Kreis der Länge höchstens $2d$ enthalten würde.

→

Da alle Knoten Grad mindestens k haben, muss dieser Baum mindestens

$$1 + k \sum_{i=1}^d (k-1)^{i-1} = 1 + k \frac{(k-1)^d - 1}{k-2} = 1 + k \frac{(k-1)^{\frac{g-1}{2}} - 1}{k-2}$$

Knoten enthalten.

- Falls g gerade $g = 2d$

Betrachte eine beliebige Kante $\{x,y\}$ in H' und alle Knoten, die von x oder y aus über höchstens $d-1$ Kanten erreichbar sind. Wieder müssen diese Knoten einen Baum bilden. Der Baum enthält mindestens

$$2 \sum_{i=1}^d (k-1)^{i-1} = 2 \frac{(k-1)^d - 1}{k-2} = 2 \frac{(k-1)^{\frac{g}{2}} - 1}{k-2}$$

Knoten.

In beiden Fällen ist die Anzahl der Knoten in H' (wie auch in H) größer als $(k-1)^{\frac{g-1}{2}}$

Also gilt

$$n > \left(\frac{m}{n} \right)^{\frac{g-1}{2}} \Rightarrow n^{\frac{2}{g-2}} > \left\lfloor \frac{m}{n} \right\rfloor \Rightarrow \left\lceil n^{\frac{2}{g-2}} \right\rceil \geq \frac{m}{n} \Rightarrow m \leq n \cdot \left\lceil n^{\frac{2}{g-2}} \right\rceil$$

□

Lemma:

Jeder Kreis in H besteht aus mehr als $t + 1$ Kanten.

Beweis: (durch Widerspruch)

Nehmen wir an, es gäbe in H einen Kreis C aus höchstens $t + 1$ Kanten.

Sei $\{u, v\}$ die letzte Kante des Kreises, die der Algorithmus in H eingefügt hat.

Der Rest des Kreises besteht aus höchstens t Kanten, deren Gewicht nicht grösser als das von $\{u, v\}$ ist. Also enthielt H vor dem Einfügen der Kante $\{u, v\}$ bereits einen Pfad von u nach v mit Gewicht höchstens $t \cdot c(\{u, v\})$, ein Widerspruch zur Definition des Algorithmus.

□

➤ Aus den letzten beiden Lemmas folgt, dass H höchstens $n \cdot \left\lceil n^{\frac{2}{t-1}} \right\rceil$ Kanten enthält.

- Sei T der minimale Spannbaum von G , der mit dem Algorithmus von Kruskal berechnet wird, wenn die Kanten in derselben Reihenfolge bearbeitet werden wie vom Algorithmus Greedy-Spanner

Algorithmus von Kruskal

Eingabe: Zusammenhängender Graph $G = (V, E)$ mit Kantengewichten $c : E \rightarrow \mathbb{R}^+$

Ausgabe: Spannbaum $T = (V, E')$ von G mit minimalem Gewicht

sortiere die Kanten in $E = \{e_1, e_2, \dots, e_m\}$ so, dass $c(e_1) \leq c(e_2) \leq \dots \leq c(e_m)$;

$E' := \Phi$;

$T := (V, E')$;

for $i = 1$ to m **do**

 // sei $e_i = \{u, v\}$

if u und v sind in verschiedenen Zusammenhangskomponenten von T **then**

$E' := E' \cup \{e_i\}$;

$T := (V, E')$;

fi

od;

return T ;

- Bezeichne mit $E(T)$ die Menge der Kanten von T und mit MST das Gewicht von T
- Bezeichne die Kanten von H weiterhin mit E'
- E' enthält alle Kanten des minimalen Spannbaums T , d.h. es gilt $E(T) \subseteq E'$, da im Algorithmus Greedy-Spanner: Falls $d_H(u, v) = \infty \rightarrow u$ und v sind in verschiedenen Zusammenhangskomponenten!!
- Sei P der Weg (Tour), den ein Präorderdurchlauf durch T zurücklegt
- Sei L die Summe der Kantengewichte von P , $L = 2 \cdot c(T) = 2 \cdot MST$
- Ordne jeden Knoten von T der Stelle auf der Tour zu, wo der Knoten zum ersten Mal besucht wird
→ Kreis, der alle Knoten in der Reihenfolge ihrer Präordernummerierung enthält und in dem der Abstand **zweier aufeinander folgender Knoten** u und v gleich $d_T(u, v)$ ist
- Es gilt $d_P(u, v) \geq d_T(u, v)$

➤ **Beispiel**

- Wir teilen die Ausführung des Algorithmus Greedy-Spanner in $\log n + 1$ Phasen ein
- Vereinfachende Annahme: Sei n eine Zweierpotenz
- Das Gewicht jeder Kante in T oder H liegt im Intervall $[0, L]$
- Wir partitionieren das Intervall $[0, L]$ in $\log n + 1$ Teilintervalle:

$$I_0 = [0, \frac{L}{n}] , \quad I_j = (2^{j-1} \cdot \frac{L}{n}, 2^j \cdot \frac{L}{n}] \quad \text{für } j = 1, 2, \dots, \log n$$

- Innerhalb einer Phase bearbeitet der Algorithmus Kanten, deren Gewichte im selben Intervall I_j liegen
- Für $0 \leq j \leq \log n$ definiere: $E_j = \{e \in E \setminus E(T) \mid c(e) \in I_j\}$
- E_j ist also die Menge derjenigen Kanten, die der Algorithmus in Phase j in $E \setminus$ einfügt und die nicht im minimalen Spannbaum T enthalten sind

Lemma: $c(E_0) \leq 2 \cdot \left\lceil n^{\frac{2}{t-1}} \right\rceil \cdot MST \leq 4 \cdot n^{\frac{2}{t-1}} \cdot MST$

Beweis:

▪ E' und damit auch E_0 besteht aus höchstens $n \cdot \left\lceil n^{\frac{2}{t-1}} \right\rceil$ Kanten (Satz (b))

▪ jede Kante in E_0 hat Gewicht höchstens $\frac{L}{n} = \frac{2}{n} \cdot MST$

▪ $\Rightarrow c(E_0) \leq n \cdot \left\lceil n^{\frac{2}{t-1}} \right\rceil \cdot \frac{2}{n} \cdot MST = 2 \cdot \left\lceil n^{\frac{2}{t-1}} \right\rceil \cdot MST$

□

Betrachtung von E_j für $j \geq 1$

- Sei $a = 2^{j-1} \cdot \frac{L}{n}$, so dass $I_j = (a, 2a]$
- Sei ein δ mit $0 < \delta < \min\{t-1, 1\}$ vorgegeben
- **Definition (Cluster):** Wir teilen den Kreis P in $\frac{2L}{\delta a}$ Intervalle der Größe $\frac{\delta a}{2}$ ein, indem

wir an einer beliebigen Stelle einen Schnitt machen und von dort aus um den Kreis

herumlaufen und nach jeweils $\frac{\delta a}{2}$ zurückgelegten Längeneinheiten wieder einen Schnitt

machen. Die resultierenden Intervalle nennen wir **Cluster**

- Sei n_j die Anzahl der Cluster, die mindestens einen Knoten aus V enthalten

$$n_j \leq \frac{2L}{\delta a} = \frac{2n}{\delta 2^{j-1}}$$

- Eine Kante $\{u, v\}$ nennen wir Intercluster-Kante, wenn u und v in verschiedenen Clustern liegen

Lemma:

Jede Kante $\{u,v\} \in E_j$ ist eine Intercluster-Kante

Beweis:

- Es gilt $\{u,v\} \notin E(T)$
- Betrachte den Pfad Q , der u und v in T verbindet
- Alle Kanten auf Q haben Gewicht höchstens $c(\{u,v\})$ (sonst wäre T kein minimaler Spannbaum) und wurden von Kruskal's Algorithmus und damit auch vom Algorithmus Greedy-Spanner vor der Kante $\{u,v\}$ bearbeitet
- Da $E(T) \subseteq E'$ gilt, muss Q bereits in H enthalten gewesen sein, als Greedy-Spanner die Kante $\{u,v\}$ bearbeitet hat
- Da $\{u,v\} \in E'$, gilt $c(Q) > t \cdot c(\{u,v\})$
- Aus $\delta < t - 1$ folgt $t > 1 + \delta$
- Damit erhalten wir: $d_P(u,v) \geq d_T(u,v) = c(Q) > t \cdot c(\{u,v\}) > (1 + \delta)c(\{u,v\}) > \delta a$
- Der Abstand zwischen zwei Knoten innerhalb eines Clusters ist höchstens $\frac{\delta a}{2}$. Also liegen u und v in verschiedenen Clustern

□

Lemma:

$$|E_j| \leq 2 \cdot n_j^{\min\{2, 1 + \frac{2+\delta}{t-1-\delta}\}}$$

Beweis:

- Da E_j nur Intercluster-Kanten enthält, gilt daher $|E_j| \leq n_j^2 \rightarrow$ z.Z. $|E_j| \leq 2 \cdot n_j^{1 + \frac{2+\delta}{t-1-\delta}}$
- Sei M der Graph, der aus (V, E_j) entsteht, indem man die Knoten jedes Clusters zu einem einzigen Knoten verschmilzt
- Sei C ein beliebiger Kreis in M , der aus g Kanten besteht. Z.Z. $g \geq \frac{t+1}{1 + \frac{\delta}{2}}$
- Seien w_1, w_2, \dots, w_g die Gewichte der g Kanten von C in aufsteigender Reihenfolge
- Als die letzte Kante (mit Gewicht w_g) in H eingefügt wurde, gab es in H bereits einen Pfad

der Länge höchstens $g \cdot \frac{\delta a}{2} + \sum_{i=1}^{g-1} w_i$ zwischen den Endknoten der Kante

$\rightarrow g \cdot \frac{\delta a}{2}$ schätzt die höchstens g Teilpfade innerhalb eines Clusters ab

$\rightarrow \sum_{i=1}^{g-1} w_i$ die Intercluster-Kanten

\rightarrow

- Da die Kante mit Gewicht w_g in H eingefügt wurde, gilt $g \cdot \frac{\delta a}{2} + \sum_{i=1}^{g-1} w_i > t \cdot w_g$
- Wegen $a \leq w_g$ und $w_i \leq w_g$ für $1 \leq i \leq g-1$ können wir folgern: $t \cdot w_g < g \cdot \frac{\delta w_g}{2} + (g-1)w_g$
- Das ergibt: $t < g\left(\frac{\delta}{2} + 1\right) - 1 \rightarrow g > \frac{t+1}{1 + \frac{\delta}{2}}$

→ Jeder Kreis in M enthält mehr als $\frac{t+1}{1 + \frac{\delta}{2}}$ Kanten

→ $|E_j| \leq 2 \cdot n_j^{1 + \frac{2+\delta}{t-1-\delta}}$

□

Lemma:

$$c(E_j) \leq \frac{32}{\delta^2} \cdot n^{\frac{2+\delta}{t-1-\delta}} \cdot MST \cdot \left(\frac{t-1}{t}\right)^{j-1}$$

Beweis:

Jede Kante in E_j hat Gewicht höchstens $2a$, daher wegen $|E_j| \leq 2 \cdot n_j^{\min\{2, 1+\frac{2+\delta}{t-1-\delta}\}}$ gilt

$$\begin{aligned} c(E_j) &\leq 2a \cdot 2 \cdot n_j^{\min\{2, 1+\frac{2+\delta}{t-1-\delta}\}} \leq 2 \cdot 2^{j-1} \frac{L}{n} \cdot 2 \cdot \left(\frac{2n}{\delta 2^{j-1}}\right)^{\min\{2, 1+\frac{2+\delta}{t-1-\delta}\}} \\ &\leq 2 \cdot 2^{j-1} \frac{L}{n} \cdot 2 \cdot \left(\frac{2}{\delta}\right)^2 \cdot \left(\frac{n}{2^{j-1}}\right)^{1+\frac{2+\delta}{t-1-\delta}} = 2 \cdot 2^{j-1} \frac{L}{n} \cdot 2 \cdot \left(\frac{2}{\delta}\right)^2 \cdot \left(\frac{n}{2^{j-1}}\right) \cdot \left(\frac{n}{2^{j-1}}\right)^{\frac{2+\delta}{t-1-\delta}} \\ &= \frac{16L}{\delta^2} \cdot \left(\frac{n}{2^{j-1}}\right)^{\frac{2+\delta}{t-1-\delta}} = \frac{16L}{\delta^2} \cdot n^{\frac{2+\delta}{t-1-\delta}} \cdot \left(\frac{1}{2^{j-1}}\right)^{\frac{2+\delta}{t-1-\delta}} = \frac{32}{\delta^2} \cdot MST \cdot n^{\frac{2+\delta}{t-1-\delta}} \cdot \left(\frac{1}{2^{j-1}}\right)^{\frac{2+\delta}{t-1-\delta}} \\ &\leq \frac{32}{\delta^2} \cdot MST \cdot n^{\frac{2+\delta}{t-1-\delta}} \cdot \left(\frac{1}{4^{t-1}}\right)^{j-1} \leq \frac{32}{\delta^2} \cdot MST \cdot n^{\frac{2+\delta}{t-1-\delta}} \cdot \left(\frac{t-1}{t}\right)^{j-1} \end{aligned}$$

□

- Mit $c(E_0) \leq 4 \cdot n^{\frac{2}{t-1}} \cdot MST$ und $c(E_j) \leq \frac{32}{\delta^2} \cdot n^{\frac{2+\delta}{t-1-\delta}} \cdot MST \cdot \left(\frac{t-1}{t}\right)^{j-1}$ können wir wegen

$E = E(T) \cup E_0 \cup E_1 \cup \dots \cup E_{\log n}$ das Gewicht $c(H)$ wie folgt abschätzen:

$$\begin{aligned} c(H) &\leq MST + 4 \cdot n^{\frac{2}{t-1}} \cdot MST + \frac{32}{\delta^2} \cdot n^{\frac{2+\delta}{t-1-\delta}} \cdot MST \cdot \sum_{j=1}^{\log n} \left(\frac{t-1}{t}\right)^{j-1} \\ &\leq MST + 4 \cdot n^{\frac{2}{t-1}} \cdot MST + \frac{32}{\delta^2} \cdot n^{\frac{2+\delta}{t-1-\delta}} \cdot MST \cdot t \\ &\leq 5 \cdot n^{\frac{2}{t-1}} \cdot MST + \frac{32t}{\delta^2} \cdot n^{\frac{2+\delta}{t-1-\delta}} \cdot MST \leq \left(5 + \frac{32t}{\delta^2}\right) \cdot n^{\frac{2+\delta}{t-1-\delta}} \cdot MST \end{aligned}$$

- Teil c) vom Satz ist somit bewiesen ☺

Korollar:

Wenn der Algorithmus Greedy-Spanner mit Parameter $t = \log_2 n$ aufgerufen wird (unter Verwendung von $\delta = 0.5$ „z.B.“), so liefert er einen t -Spanner H mit $O(n)$ Kanten und $c(H) = O(\log n) \cdot MST$

Modellierung eines Netzwerkdesign-Problems

- Gegeben ist eine Menge V von n Knoten, die vernetzt werden sollen
- Die Knoten sind von 1 bis n durchnummeriert
- Für jedes Knotenpaar (i, j) ist eine **Verkehrsanforderung** $r_{i,j}$ gegeben, die bestimmt, wie viel Verkehr pro Zeiteinheit von i nach j geschickt werden soll
- $r_{i,j}$ als Vielfache einer gewissen Basis-Einheit (z.B. 2 Mbps) gegeben (nichtganzzahlige Vielfache zugelassen)
- Lösung: Welche Links mit welcher Kapazität zu installieren? ← Routing der Anforderungen (Pfade)?
- Kapazität jedes Links \geq Summe aller durch ihn gerouteten Anforderungen!!!
- Lösungskosten: Summe der Kosten jedes Links
- Linkskosten: zwei Komponenten
 - Kapazitätsunabhängige Grundkosten F (für jeden Link gleich)
 - Kapazitätsabhängige Kosten (abhängig von **Linkslänge** ($p_{i,j}$) und **Kapazität** ($u_{i,j}$)
„ganzzahliges Vielfaches der Basis-Einheit“) $\rightarrow p_{i,j}u_{i,j}$
- \rightarrow Gesamtkosten $F + p_{i,j}u_{i,j}$
- Jede Anforderung $r_{i,j}$ wird entlang einem Pfad $P_{i,j}$ geroutet

Modellierung eines Netzwerkdesign-Problems

- Für jede Kante e bezeichne mit $q(e)$ die Summe aller Anforderungswerte $r_{i,j}$, deren Pfad $P_{i,j}$ die Kante e enthält
- Links e mit $q(e) > 0$ sind einzurichten mit Kapazität (am günstigsten) $\lceil q(e) \rceil$
- Bezeichne mit E' für ein Routing die Menge der Kanten e mit $q(e) > 0$
- Das so modellierte Problem nennen wir „Netzwerkdesign mit einem Kabeltyp“

Problem Netzwerkdesign mit einem Kabeltyp (ND1K)

Instanz: Menge V von n Knoten, Anforderungsmatrix $R = (r_{i,j})$, Fixkosten F ,
Preismatrix $P = (p_{i,j})$

Lösung: Pfad $P_{i,j}$ für jede Anforderung $r_{i,j} > 0$, $1 \leq i < j \leq n$

Ziel: minimiere $\sum_{e=\{i,j\} \in E'} (F + \lceil q(e) \rceil \cdot p_{i,j})$

- **Annahmen:** - das Netz ist zusammenhängend \rightarrow die Optimallösung enthält min. $n-1$ Kanten
- die Matrizen R und P sind symmetrisch
- für die Preise $p_{i,j}$ gilt die Dreiecksungleichung d.h. $p_{i,j} \leq p_{i,k} + p_{k,j}$ für alle k
- Wir betrachten das geplante Netz als ungerichteten Graphen und die Verkehrsströme als ungerichtete Pfade

Untere Schranken für die Kosten OPT einer optimalen Lösung

Lemma:

(a) $OPT \geq F(n-1) + MST$, mit Kantengewichten $p_{i,j}$

(b) $OPT \geq F(n-1) + \sum_{1 \leq i < j \leq n} r_{i,j} \cdot p_{i,j}$

Beweis:

- (a) - Das (zusammenhängende) Netz enthält einen Spannbaum, in dem jede Kante Kapazität mindestens 1 hat
- Die **kapazitätsunabhängigen** Kosten für die $n-1$ Kanten in diesem Spannbaum sind $F(n-1)$
 - Die **kapazitätsabhängigen** Kosten sind mindestens MST , da kein Spannbaum mit Kantenkapazität 1 weniger kapazitätsabhängige Kosten haben kann als der minimale Spannbaum
- (b) Jede Anforderung $r_{i,j}$ verursacht min. Kapazitätsabhängige Kosten $r_{i,j} \cdot p_{i,j}$, da kein Pfad von i nach j eine Länge kleiner als $p_{i,j}$ haben kann (Dreiecksungleichung)

□

- Eine Variante von ND1K, wo alle Anforderungen einen ausgezeichneten Knoten (z.B. 1) mit einem anderen Knoten verbinden
- Nenne den Knoten 1 Quelle und das sich ergebende Problem **ND1K mit einer Quelle**

Definition:

Problem ND1K mit einer Quelle

Instanz: Menge V von n Knoten, Anforderungen $r_{1,j}$, $2 \leq j \leq n$, Fixkosten F ,
Preismatrix $P = (p_{i,j})$

Lösung: Pfad $P_{1,i}$ für jede Anforderung $r_{1,i}$, $2 \leq i \leq n$

Ziel: minimiere $\sum_{e=\{i,j\} \in E} (F + \lceil q(e) \rceil \cdot p_{i,j})$

Achtung: $r_{1,i} > 0$ für alle $2 \leq i \leq n$, deshalb muss die Quelle mit **allen** anderen Knoten verbunden werden!!

Algorithmus: Netzwerkdesign mit einer Quelle

Eingabe: Menge V mit n Knoten (Knoten 1 ist die Quelle), Anforderungen $r_{1,i}$ für $2 \leq i \leq n$, Fixkosten F , Preismatrix $P = (p_{i,j})$

Ausgabe: Pfade $P_{1,i}$ für $2 \leq i \leq n$

1. berechne $(1 + \sqrt{2}, 1 + \sqrt{2})$ -LAST T mit Startknoten 1 in dem vollständigen Graphen mit Knotenmenge V und Kantengewichten $p_{i,j}$;
2. $P_{1,i} :=$ der Pfad von 1 nach i in T (für alle $i = 2, \dots, n$);

Approximationsrate des Algorithmus

Satz:

Der Algorithmus liefert ein Routing, bei dem die Kosten für das Netz höchstens $(2 + 2\sqrt{2}) \cdot OPT$ sind. Der Algorithmus ist somit ein $(2 + 2\sqrt{2})$ -Approximationsalgorithmus für ND1K mit einer Quelle

Beweis:

- Bezeichne mit E' die Menge der Kanten des (α, β) -LAST T , die der Algorithmus in Schritt 1 berechnet
- Jede dieser Kanten ist am Ende auch in mindestens einem der berechneten Pfade $P_{l,i}$ enthalten
- definiere $\Delta(e) = \lceil q(e) \rceil - q(e)$: der Teil der Kapazität von e , den wir zwar bezahlen müssen, den wir aber eigentlich nicht benötigen
- Die Kosten des Netzes, das sich aus dem Routing des Algorithmus ergibt, sind

$$C = F \cdot (n-1) + \sum_{e=\{i,j\} \in E'} \lceil q(e) \rceil \cdot p_{i,j}$$

$$\text{▪ } C = A + B \quad A = \sum_{e=\{i,j\} \in E'} q(e) \cdot p_{i,j}$$

$$B = F \cdot (n-1) + \sum_{e=\{i,j\} \in E'} \Delta(e) \cdot p_{i,j}$$

→

$$A = \sum_{e=\{i,j\} \in E} q(e) \cdot p_{i,j}$$

- Der Beitrag einer Anforderung $r_{1,i}$ zu der Summe für A ist $r_{1,i} d_T(1,i)$, wobei $d_T(1,i)$ die Länge des Pfades von 1 nach i in T bezüglich der Kantengewichte $p_{j,k}$ ist

$$\rightarrow A = \sum_{i=2}^n r_{1,i} d_T(1,i)$$

- Da T ein (α, β) -LAST ist, gilt $d_T(1,i) \leq \alpha d_G(1,i) = \alpha p_{1,i}$ (Dreiecksungleichung)

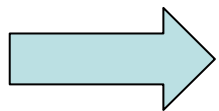
$$\rightarrow A \leq \alpha \sum_{i=2}^n r_{1,i} \cdot p_{1,i}$$

- Mit $OPT \geq F \cdot (n-1) + \sum_{1 \leq i < j \leq n} r_{i,j} \cdot p_{i,j}$ erhalten wir $A \leq \alpha OPT$

→

$$B = F \cdot (n-1) + \sum_{e=\{i,j\} \in E} \Delta(e) \cdot p_{i,j}$$

- Da $\Delta(e) < 1 \rightarrow B \leq F \cdot (n-1) + \sum_{e=\{i,j\} \in E} p_{i,j}$
- Da T ein (α, β) -LAST ist, gilt $\sum_{e=\{i,j\} \in E} p_{i,j} \leq \beta \cdot MST$
- Mit $OPT \geq F(n-1) + MST$, folgt $B \leq \beta OPT$
- $\rightarrow C = A + B \leq (\alpha + \beta)OPT$
- Da $\beta = 1 + \frac{2}{\alpha - 1}$, wird $\alpha + \beta$ minimal, wenn $\alpha = \beta = 1 + \sqrt{2}$



Approximationsrate $2 + 2\sqrt{2} \approx 4.828$

□

Die allgemeine Version des Problems ND1K

Algorithmus: Netzwerkdesign

Eingabe: Menge V von n Knoten, Anforderungsmatrix $R = (r_{i,j})$,

Fixkosten F , Preismatrix $P = (p_{i,j})$

Ausgabe: Pfade $P_{i,j}$ für alle $1 \leq i < j \leq n$ mit $r_{i,j} > 0$

1. berechne mit Algorithmus Greedy-Spanner einen $\log n$ -Spanner H in dem vollständigen Graphen mit Knotenmenge V und Kantengewichten $p_{i,j}$;
2. $P_{i,j} :=$ ein kürzester Pfad von i nach j in H (für alle $1 \leq i < j \leq n$ mit $r_{i,j} > 0$)

Satz:

Der Algorithmus liefert für Probleme mit n Knoten ein Routing, bei dem die Kosten für das Netz höchstens $O(\log n) \cdot OPT$ sind. Der Algorithmus ist somit ein $O(\log n)$ -Approximationsalgorithmus für ND1K

Beweis:


- Bezeichne mit \tilde{E} die Menge der Kanten des $\log n$ -Spanners H , den der Algorithmus in Schritt 1 berechnet
- Sei $E' \subseteq \tilde{E}$ die Menge der Kanten, die am Ende in mindestens einem der berechneten Pfade $P_{i,j}$ enthalten sind
- Die Kosten des Netzes, das sich aus dem Routing des Algorithmus ergibt, sind

$$C = F \cdot |E'| + \sum_{e=\{i,j\} \in E'} \lceil q(e) \rceil \cdot p_{i,j}$$

- $C = A + B$
 $A = \sum_{e=\{i,j\} \in E'} q(e) \cdot p_{i,j}$
 $B = F \cdot |E'| + \sum_{e=\{i,j\} \in E'} \Delta(e) \cdot p_{i,j}$

→

Allgemeines Netzwerkdesign

- $A = \sum_{1 \leq i < j \leq n} r_{i,j} \cdot d_H(i, j)$ wobei $d_H(i, j)$ die Länge eines kürzesten Pfades von i nach j in H bezüglich der Kantengewichte $p_{i,j}$ ist
 - Da H ein $\log n$ -Spanner von G ist, gilt $d_H(i, j) \leq (\log n) d_G(i, j) = (\log n) p_{i,j}$ (Dreiecksungl.)
 - $\rightarrow A \leq (\log n) \cdot \sum_{1 \leq i < j \leq n} r_{i,j} \cdot p_{i,j}$
 - Mit $OPT \geq F \cdot (n-1) + \sum_{1 \leq i < j \leq n} r_{i,j} \cdot p_{i,j}$, erhalten wir $A \leq \log n \cdot OPT$
 - Da $\Delta(e) < 1$ gilt, folgt $B \leq F \cdot |E'| + \sum_{e=\{i,j\} \in E'} p_{i,j}$
 - $|E'| \leq |\tilde{E}| = O(n)$ (!Korollar!) $\rightarrow F \cdot |E'| \leq c F (n-1)$ für eine Konstante c
 - $\sum_{e=\{i,j\} \in E'} p_{i,j} \leq \sum_{e=\{i,j\} \in \tilde{E}} p_{i,j} = O(\log n) \cdot MST$ (!Korollar!)
 - Mit $OPT \geq F (n-1) + MST$, folgt $B \leq O(\log n) \cdot OPT$
-  $C = A + B \leq O(\log n) \cdot OPT$

□

Vielen Dank für die Aufmerksamkeit 😊

Fragen?

