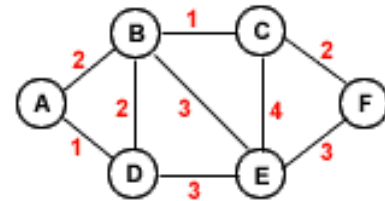




Effiziente Algorithmen

Hartmut Klauck
Universität Frankfurt
SS 06
18.5.





Ungerichtete Graphen

- Unser Ziel ist es, für ungerichtete Graphen paarweise Distanzen in Zeit $O(n^2 \text{ polylog } n)$ mit Verlustfaktor 3 zu approximieren
- Der Algorithmus benutzt daher nicht Matrixmultiplikation
- Dies macht den Algorithmus wesentlich praktikabler als Algorithmen, die Matrixmultiplikation benutzen
- Erst Approximation mit Verlustfaktor ≥ 2 und die Einschränkung auf ungerichtete Graphen machen dies möglich

Die Partitionierungsprozedur

- Eingabe Graph G, W , Folge $s_1 > \dots > s_{k-1}$ (k ist ein Parameter)
- 1. Für $i=2$ bis k setze $E_i = \{e \in E : \text{ind}(e) \leq s_{i-1}\}$
 - Bemerkung: wenn ein Knoten v einen Grad kleiner als s_{i-1} hat, werden alle Kanten gewählt
- 2. Für $i=1$ bis $k-1$ setze $(D_i, E_i^*) = \text{dominate}(V, E_{i+1}, s_i)$
- 3. $E_1 = E, D_k = V, E^* = \bigcup E_i^*$
 - Bemerkung: $|E^*| \leq kn$



Eine 3-Approximation

- Diesmal werden wir den Graphen in $\log n$ Mengen aufteilen
- Dann führen wir $\log n$ mal Dijkstra von Knotenmengen aus durch

Algorithmus stretch3

- Eingaben: Graph G , Gewichte W
- Ausgabe: Matrix M von geschätzten Distanzen, mit $\delta(u,v) \leq M(u,v) \leq 3\delta(u,v)$
- 1. Setze $s_i = n/2^i$ für $i=1, \dots, k = \log n$
- 2. $(E_1, E_2, \dots, E_k, E^*, D_1, \dots, D_k) = \text{partition}(G, W, s_1, \dots, s_{k-1})$
- 3. Für alle u, v setze $M(u,v) = W(u,v)$
- 4. Für $i=1, \dots, k$:
 - Für jedes u aus D_i :
 $\text{dijkstra}(V, E_i \cup E^* \cup (\{u\} \times V), M, u)$

Laufzeit

- partition läuft in $O((n+m) \log n)$
- Schritt 4: für jedes i von 1 bis $k = \log n$ wird dijkstra $O((n \log n)/s_i)$ mal auf Graphen mit $O(n s_{i-1})$ Kanten ausgeführt.
- Laufzeit dafür

$$\begin{aligned} & O\left(\sum_{i=1}^{\log n} n \log h / s_i \cdot (n + n \cdot s_{i-1}) \cdot \log n\right) \\ &= O\left(\sum_{i=1}^{\log n} \log^2 n \cdot 2^i \cdot h^2 / 2^{i-1}\right) \\ &= O\left(h^2 \cdot \log^2 n \cdot \sum_{i=1}^{\log n} 2^i / 2^{i-1}\right) = O(h^2 \log^3 n) \end{aligned}$$



Korrektheit

- Bezeichne $M_i(u,v)$ die Distanzschätzung nach i Runden in Schritt 4
- Für Pfad p sei $W(p)$ sein Gewicht
- Wir zeigen:
 - **Lemma:** Sei p ein Pfad, der u aus D_i und v aus V verbindet. Dann gibt es eine Menge von Kanten $A_i(p)$ auf p , so daß $|A_i(p)| \leq \underline{i-1}$, $A_i(p) \cap E_i = \emptyset$ und $M_i(u,v) \leq W(p) + 2W(A_i(p))$.



Korrektheit

- **Theorem:** Der Algorithmus berechnet Distanzen M mit $\delta(u,v) \leq M(u,v) \leq 3\delta(u,v)$
- **Beweis:** Durch die Dijkstra Algorithmen wird nie eine zu kleine Schätzung $M(u,v)$ ausgegeben.
- Sei p ein kürzester Pfad u nach v , d.h. $W(p) = \delta(u,v)$
- $M(u,v) = M_k(u,v)$, $D_k = V$, daher nach dem Lemma $M(u,v) \leq 3 W(p) \leq 3\delta(u,v)$



Beweis des Lemmas

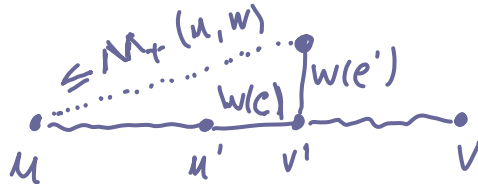
- **Lemma:** Sei p ein Pfad, der u aus D_i und v aus V verbindet. Dann gibt es eine Menge von Kanten $A_i(p)$ auf p , so daß $|A_i(p)| \leq i-1$, $A_i(p) \cap E_i = \emptyset$ und $M_i(u,v) \leq W(p) + 2W(A_i(p))$.
- Induktion über i
- Für alle u aus D_1 und alle v gilt $M_1(u,v) = \delta(u,v)$, also gilt die Behauptung für $i=1$
- Die Behauptung gelte für alle $1 \leq t < i$
- p verbinde u aus D_i und v aus V
- Wenn alle Kanten aus E_i zu p gehören, gilt $M_i(u,v) \leq W(p)$ [d.h. wir sind fertig]
- Ansonsten sei $e=(u',v')$ die letzte Kante auf p , die nicht in E_i liegt

Beweis des Lemmas

- Sei t so, dass $e \in E_t$ und e nicht in E_{t+1}
- Es gibt nun ein $e'=(v'.w)$ aus $E_{t+1} \cap E^*$ so dass $w \in D_t$ und $W(e') \leq W(e)$
- Sei p' der folgende Pfad: wie p von u nach v' , dann per e' nach w
- Per Induktion (für Pfad p' von $w \in D_t$ nach u) gilt:
 $M_t(u,w)=M_t(w,u)$
 $\leq W(p')+2W(A_t(p')) = W(p_{u,v'})+W(e')+ 2W(A_t(p'))$
- Dabei ist $A_t(p')$ von der Grösse $t-1$ und disjunkt von E_t
- e' liegt in $E_{t+1} \subseteq E_t$, also nicht in $A_t(p')$, somit liegen alle Kanten von $A_t(p')$ in $p_{u,v'}$
- $e \in E_t$, daher ist e nicht in $A_t(p')$

Beweis des Lemmas

- In Runde i wird von u aus dijkstra ausgeführt, und die Kantenmenge im dijkstra Lauf enthält eine Kante (u,w) mit Gewicht $\leq M_+(u,w)$
- Die Kantenmenge im dijkstra Lauf enthält ausserdem $e' \in E^*$ und alle Kanten des Pfades p von v' bis v



- Daher gilt $M_i(u,v) \leq M_+(u,w) + W(e') + W(p_{v',v})$
 $\leq W(p_{u,v'}) + W(p_{v',v}) + 2W(e') + 2W(A_+(p'))$
 $\leq W(p_{u,v}) + 2W(e) + 2W(A_+(p'))$
- Wir setzen $A_i(p) = A_+(p') \cup \{e\}$, und die Behauptung folgt.
- Dies ist zulässig, da $A_+(p')$ nur Kanten aus $p_{u,u'}$ enthält.



Zusammenfassung

- In Zeit $O(n^2 \log^3 n)$ erhalten wir eine 3-Approximation
- Der Algorithmus ist einfach und hat gute Laufzeitkonstanten
- Für ungewichtete Graphen gilt:
der Algorithmus gibt Distanzen aus, die um $2|A_k(p)| \leq 2k=2 \log n$ *additiv* danebenliegen können



Zusammenfassung

- Wir haben das APSP Problem untersucht (für den Fall nichtnegativer Kantengewichte)
- Für ungerichtete ungewichtete Graphen konnten wir randomisiert in ungefähr der Laufzeit der Matrixmultiplikation exakt rechnen, bei gewichteten, gewichteten Graphen ging dies bei einer $1+\varepsilon$ -Approximation
- Bessere Laufzeit lässt sich bei gerichteten Graphen nicht erzielen, selbst bei schlechterer Approximation
- Für ungerichtete gewichtete Graphen in fast Linearzeit bei 3-Approximation



Techniken

- Greedy Algorithmen: Dijkstra
- Dynamisches Programmieren: Floyd Warshall
- Randomisierung
 - Sampling
 - Isolierung von Zeugen
- Approximation
 - beliebig dicht
 - mit konstantem Faktor
- Reduktionen
 - von und zur Matrixmultiplikation