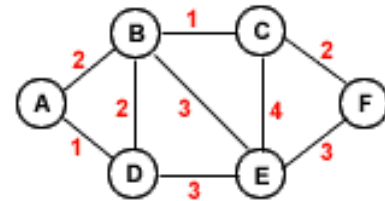




Effiziente Algorithmen

Hartmut Klauck
Universität Frankfurt
SS 06
11.5.





Zusammenfassung:

- Theorem

- Für gerichtete Graphen mit Kantengewichten aus $\{1, \dots, M\}$ kann das APD Problem mit Approximationqualität $(1+\varepsilon)$ in Zeit

$O(T(n) \cdot \log(Mn) \cdot \log^4 n \log \log n / \varepsilon)$
für die Laufzeit $T(n)$ der Integermatrixmultiplikation gelöst werden.



Frage:

- Können wir eine solche Approximation oder sogar eine exakte Berechnung der paarweisen Distanzen auch noch schneller erzielen?



Antwort: Nein!

- **Theorem:** Wenn ein Algorithmus in Zeit $K(n)$ für ungerichtete, ungewichtete Graphen die paarweisen Distanzen mit Verlustfaktor < 2 approximiert, kann das Boolesche Matrixprodukt in Zeit $O(K(n))$ berechnet werden.
- Kommentar: Im wesentlichen sind die Probleme (Boolesche) Matrixmultiplikation und sehr gute Approximation von APD also äquivalent.
- Das Resultat zeigt in gewissem Sinne, dass Approximation von APD schwer ist, per Reduktion ähnlich wie in der NP-Vollständigkeitstheorie.
- Wenn Boolesche Matrixmultiplikation nicht in Linearzeit berechenbar ist, dann APD auch nicht, nicht einmal bei $2-\varepsilon$ Approximation



Allerdings

- Wir werden nachher sehen, wie APD auf ungerichteten, gewichteten Graphen mit Verlustfaktor 3 in Zeit $O(n^2 \text{ polylog } n)$ approximiert werden kann.



Beweis

- Seien also A und B Boolesche $n \times n$ Matrizen
- Wir wollen einen Graphen G konstruieren, so dass wir aus den paarweisen Distanzen das Boolesche Produkt AB „ablesen“ können
- G ist wie folgt definiert:
 - $V = \{u_1, \dots, u_n\} \cup \{v_1, \dots, v_n\} \cup \{w_1, \dots, w_n\}$
 - $E = \{(u_i, v_k) : A[i, k] = 1\} \cup \{(v_k, w_j) : B[k, j] = 1\}$



Beweis

- Sei $C=AB$
- Klar: $C[i,j]=1$ gdw $\delta(u_i, w_j)=2$
- Ausserdem: $\delta(u_i, w_j) \geq 4$ wenn $C[i,j]=0$
- Wenn wir also $\delta(u_i, w_j)$ mit Verlustfaktor < 2 approximieren können, können wir auch C berechnen.



Eine weitere Beobachtung

- **Theorem:** Wenn ein Algorithmus in Zeit $K(n)$ für gerichtete, ungewichtete Graphen die paarweisen Distanzen mit einem *endlichen* Verlustfaktor approximiert, kann das Boolesche Matrixprodukt in Zeit $O(K(n))$ berechnet werden.
- **Kommentar:** Unser Algorithmus aus der vorigen Vorlesung ist also im wesentlichen optimal, keine Approximation mit endlichem Verlustfaktor kann seine Laufzeit wesentlich schlagen.
- Approximation scheint bei ungerichteten Graphen besser zu wirken



Beweis:

- Betrachte selbe Konstruktion für Graphen wie zuvor, allerdings richte Kanten von u nach v und von v nach w
- Jetzt: $C[i,j]=1$ gdw $\delta(u_i, w_j)=2$
- $C[i,j]=0$ gdw $\delta(u_i, w_j)=\infty$



Ungerichtete Graphen

- Unser Ziel ist es, für ungerichtete Graphen die paarweisen Distanzen in Zeit $O(n^2)$ 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



Einige Grundlagen

- Wir betrachten Graphen $G=(V,E)$ mit beliebigen nichtnegativen Gewichten
- Eine dominierende Menge für eine Menge $V' \subseteq V$ ist eine Teilmenge $D \subseteq V$, so dass alle Knoten in V' mindestens einen Nachbarn in D haben
- Eine minimale dominierende Menge zu finden ist NP-schwierig.
- Folgendes aber geht:
- **Theorem:** Sei $1 \leq s \leq n$. Wir können in Zeit $O(m+n)$ eine Menge D bestimmen, die alle Knoten vom Grad mindestens s dominiert, und $|D|=O(n \log n/s)$. Der Algorithmus ist randomisiert.



Algorithmus für D

- Annahme: $s \geq 10 \log n$, sonst setze $D=V$
- Füge jeden Knoten v mit Wahrscheinlichkeit $(2 \log n)/s$ in die Menge D ein
- Dann: erwartete Anzahl von Knoten ist $O(n \log n / s)$
- Zu zeigen: Die Menge der Knoten mit Grad s wird mit hoher Wahrscheinlichkeit dominiert.
- Sei also v ein Knoten mit Grad $t \geq s$
- Jeder seiner Nachbarn hat Ws. $(2 \log n)/s$, in D eingefügt zu werden
- Die erwartete Anzahl der Nachbarn von v in D ist also $t \cdot 2 \log n / s \geq 2 \log n$
- Wie gross ist die Wahrscheinlichkeit, weniger als 1 Nachbarn in D zu haben?
- Dies ist die Wahrscheinlichkeit, $t \geq s$ mal ein Ereignis mit Ws. $(1-2 \log n/s)$ zu erhalten, also beschränkt durch $(1-2 \log n/s)^s \leq (1/e)^{2 \log n} \leq 1/n^2$



Algorithmus für D

- Für jedes v mit $\text{Grad } t \geq s$ ist also die Wahrscheinlichkeit, keinen Nachbarn in D zu haben, kleiner als $1/n^2$
- Damit gilt:
 $\text{Prob}(\exists v \in V, \text{deg}(v) \geq s \text{ und } v \text{ hat keinen Nachbarn in } D)$
 $\leq n \cdot \text{Prob}(\text{deg}(v) \geq s \text{ und } v \text{ hat keinen Nachbarn in } D)$
 $\leq 1/n$
- Somit haben wir eine hohe Wahrscheinlichkeit, eine nicht zu grosse, Grad s dominierende Menge zu finden
- Dieses Ergebnis kann auch durch einen deterministischen Algorithmus erzielt werden



Zusammenfassung

- Wir erhalten einen randomisierten Algorithmus mit erwarteter Laufzeit $O(m+n)$, der eine Knotenmenge D bestimmt, die $O((n \log n)/s)$ Knoten enthält und alle Knoten mit Grad s dominiert.
- (Wir verwenden, dass D in Zeit $O(m+n)$ getestet werden kann).
- Prozedur $\text{dominate}(G, s)$ bestimme eine solche Menge D sowie eine Menge von Kanten E^* , die für alle Knoten mit Grad s eine Kante zu einem Knoten in D enthält.