Seminar über Kryptographie und Komplexität
Wintersemester 1996/1997

Divide-and-Conquer Approximation Algorithms via Spreading Metrics

Guy Even Joseph Naor Satish Rao
Baruch Schieber

Vortrag von Frank Pilhofer

12. Februar 1997

1 Einleitung

Es wird ein allgemeiner Algorithmus zum näherungsweisen Lösen von Optimierungsproblemen auf Graphen vorgestellt. Als Ansatz wird das Prinzip des *Divide and Conquer* angewendet: Das ursprüngliche Problem wird in mehrere Teilprobleme zerlegt, welche anschließend unabhängig voneinander gelöst werden. Durch die unabhängige Behandlung der Teilprobleme entstehen Fehler, die abschätzbar sein müssen.

$$\mathrm{cost}(G) \leq \begin{cases} 0 & \text{falls Problem trivial} \\ \sum_i \mathrm{cost}(G_i) + \mathrm{cost}(\mathrm{cut}) & \text{sonst} \end{cases}$$

Die Lösung des Gesamtproblems kann durch die Größe der Teilprobleme approximiert werden, wenn auch der „Teilungsfehler“ abgeschätzt werden kann.

Ein Beispiel für die Anwendung des Algorithmus ist das Problem des Linear Arrangement, bei dem die Knoten eines Graphen unter Minimierung der Summe der Kantenlängen auf äquidistante, linear angeordnete Punkte abgebildet werden. Das Problem ist NP-hart.

Es läßt sich zeigen, daß mittels Divide and Conquer die optimale Lösung bis auf $O(\log n)$ approximiert werden kann, sofern der Graph in jedem Schritt unter Teilung *minimal vieler* Kanten in nichttriviale Teile zerlegt wird. Das „min-cut“-Problem ist aber ebenfalls NP-Hart.

Der Algorithmus von Leighton und Rao findet in polynomineller Zeit eine $O(\log n)$-Approximation des min-cut. Da der Teilungsschritt bei n Knoten $O(\log n)$ mal wiederholt wird, ist so eine $O(\log^2 n)$-Approximation von Linear Arrangement möglich.

Der hier vorgestellte Algorithmus verbessert dieses Ergebnis und berechnet in Polynominalzeit eine $O(\log n \log\log n)$-Approximation.

2 Idee

Der zuvor vorgestellte Algorithmus beschränkt sich auf das Suchen des mincuts, ohne aber die Größe der entstandenden Teilprobleme zu berücksichtigen. Im Sinne der obigen Rekursionsformel kann der ungünstige Fall eintreten, daß die Teilprobleme relativ klein sind und die Abschätzung des „Teilungsfehlers“ die Approximation verschlechtert.

Die Idee zur Verbesserung besteht darin, den Teilungsfehler gegen die Größe der Teilprobleme abzuwägen: sind unsere Teilprobleme groß, so können wir uns auch einen großen Teilungsfehler erlauben. Auf diese Weise können die „Teilungskosten“ mit den Kosten der Teilprobleme so verrechnet werden, daß eine bessere Approximation erreicht wird. Übertrieben gesagt werden die Teilungskosten in den Kosten der Teilprobleme „untergehen“.

Benötigt wird aber ein Maß für die Größe eines (Teil-)Problems, das über eine „Spreizmetrik“ (spreading metric) gewonnen wird.

3 Definitionen

Definition 1 (Spreizmetrik) Eine Spreizmetrik ist eine Funktion auf der Kantenmenge E eines Graphen $G = (V, E)$ und ordnet jeder Kante $e \in E$ eine rationale Länge $l(e) \geq 0$ zu.

Definition 2 (Volumen) Das Volumen einer Spreizmetrik ist definiert als die Summe aller Kantenlängen, $\mathrm{vol}(G) = \sum_{e \in E} l(e)$.

Definition 3 (Abstand) Der Abstand $\mathrm{dist}(u, v)$ zweier Punkte u und v gemäß einer Spreizmetrik ist definiert als der kürzeste Weg von u nach v entsprechend der zugeordneten Kantenlängen.

Für unser Problem fordern wir

- Die Spreizmetrik muß für jede Probleminstanz in polynomineller Zeit berechnet werden können.
- Das Volumen der Spreizmetrik sei eine untere Schranke für die optimale Lösung unseres Problems, $\mathrm{vol}(G) \leq \mathrm{cost}^*(G))$
- In jedem zusammenhängenden Teilgraphen $H = (V_H, E_H) \subseteq G$ gibt es zwei Punkte u, v mit $\mathrm{dist}(u, v) \geq |V_H| - 1$.

Somit haben wir die „Kosten“ unseres Optimierungsproblems in das „Volumen“ des Graphen übersetzt.

Unser Algorithmus wird von einem Punkt ausgehend eine Kreisumgebung wachsen lassen, bis die Anzahl der durchgeschnittenen Kanten und das vom Kreis eingeschlossene Volumen des Graphen in einem gewünschten Verhältnis stehen. Dieser Arbeitsschritt ist die „Cut-Prozedur“ und benötigt als Vorarbeit noch ein paar Definitionen.

Definition 4 (Cut) Der Cut $\mathrm{cut}(u, r)$ schneidet alle Kanten, die sich von u ausgehend im Abstand r befinden und teilt den Ursprungsgraphen so in zwei oder mehr Teilgraphen. Der Graph $C = (V_C, E_C) \subseteq G$ ist die Beschränkung von $G = (V, E)$ auf alle Punkte innerhalb des Cuts $C = G \cap \{v \in V | \mathrm{dist}(u, v) < r\}$.

Das Volumen eines Cuts definieren wir, wie zuvor, als die Summe der Kantenlängen des so definierten Graphen C, aber *plus* dem Anteil der durchgeschnittenen Kanten, der noch innerhalb der Kreisumgebung des Cuts liegt.

Definition 5 (Volumen des Cuts) Das Volumen eines Cuts $\mathsf{cut}(u,r)$, $\mathsf{vol}(C)$ ist definiert als

$$\mathsf{vol}(C) = \sum_{e \in E_C} l(e) + \sum_{\{e=(x,y)\in E \mid x \in V_C, y \notin V_C\}} (r - \mathsf{dist}(u,x))$$

Definition 6 (Kapazität des Cuts) Die Kapazität eines Cuts $c(C)$ ist definiert als die Anzahl der von dem Cut durchgeschnittenen Kanten.

4 Algorithmus

Der zentrale Satz, der unsere Approximation ermöglicht, ist das folgende „Cut-Lemma". Es garantiert die Existenz eines Cuts, der unseren Bedingungen genügt: die Anzahl der durchschnittenen Kanten (Kapazität) und das Volumen werden gegeneinander abgewägt.

Sei $G = (V, E)$ und seien u, v Punkte mit Abstand $\Delta = \text{dist}(u, v) \geq |V| - 1$. Die Existenz solcher zwei Punkte ist aufgrund unserer Vorbedingungen garantiert.

Satz 1 (Cut-Lemma) Dann gibt es einen Radius $\frac{\Delta}{4} \leq r \leq \frac{\Delta}{2}$, so daß für den so definierten Cut $C = \mathsf{cut}(u,r)$ gilt:

$$c(C) \leq \frac{8\mathsf{vol}(C)}{|V| - 1} \ln\left(\frac{4\mathsf{vol}(G)}{\mathsf{vol}(C)}\right) \ln\ln\left(8\mathsf{vol}(G)\right)$$

Der Teilungsschritt ist effizient durchführbar. Die speziellen Knoten u, v können in Polynominalzeit mit dem „all-pairs shortest path"-Algorithmus gefunden werden, der die kürzesten Wege zwischen allen Knoten berechnet. Anschließend wird von u ein „single source shortest path"-Algorithmus gestartet, bis die gewünschte Bedingung zutrifft.

Die Einschränkung, die der Satz für den Radius des Cuts liefert, stellt sicher, daß der Graph in nichttriviale Teile zerlegt wird. Eine Rekursion wird also $O(\log n)$ Schritte benötigen.

Bei Linear Arrangement wissen wir, daß jede durchschnittene Kante maximal einen Fehler von $|V|-1$ verursacht, der Teilungsfehler unseres Cuts ist daher $\text{cost(cut)} \leq c(C) \cdot (|V| - 1)$ und kann mit obigem Ergebnis abgeschätzt werden. Wir denken uns diesen Fehler als über den gesamten Teilgraphen C verteilt, und ordnen jeder „Volumeneinheit" des Teilgraphen den Fehler $\frac{\text{Fehler(cut)}}{\text{vol}(C)}$ zu.

$$\frac{\text{Fehler}}{\text{Volumeneinheit}}(C) \leq 8 \ln\left(\frac{4\text{vol}(G)}{\text{vol}(C)}\right) \ln\ln\left(8\text{vol}(G)\right)$$

Führen wir nun einen zweiten Rekursionsschritt auf dem Graphen C durch, und schneiden ihm nach dem Cut-Lemma einen Untergraphen C' ab. Wie zuvor fällt hier ein Teilungsfehler an, den wir wiederum auf alle Volumeneinheiten des Graphen C' verteilen.

$$\frac{\text{Fehler}}{\text{Volumeneinheit}}(C') \leq 8\ln\left(\frac{4\text{vol}(C)}{\text{vol}(C')}\right)\ln\ln\left(8\text{vol}(C)\right)$$

Den Volumeneinheiten von C' haben wir nun schon zum zweiten Mal Fehler zugeordnet, sowohl beim ersten als auch beim zweiten Teilungsschritt: zählen wir beide Fehler zusammen, so sehen wir, daß sich im einfach logarithmischen Term das Volumen von C herauskürzt!

Bei fortgesetzter Rekursion kann einer Volumeneinheit des ursprünglichen Graphen C im ungünstigsten Fall in *jedem* Schritt ein weiterer Fehler zugeordnet werden. Insgesamt hat sich dort dann ein Fehler von

$$\frac{\text{Fehler}}{\text{Volumeneinheit}}(\text{Gesamt}) \leq 8\ln\left(\frac{4\text{vol}(G)}{\text{vol}(C^*)}\right)\ln\ln\left(8\text{vol}(G)\right)$$

akkumuliert, wobei C^* ein Graph ist, der im letzten Teilungsschritt in triviale Graphen zerfällt. Aufgrund unserer Voraussetzungen ist $\text{vol}(C^*) \geq \frac{1}{4}$. (Enthält ein Graph zwei Knoten und eine Kante, so hat diese eine Länge größer 1 und wird nach Cut-Lemma zwischen $\frac{1}{4}$ und $\frac{1}{2}$ durchgeschnitten.)

Alle anderen Volumeneinheiten des Teilgraphen C, denen nicht in jedem Schritt ein Fehler zugeordnet wurde, haben dementsprechend *weniger* Fehler akkumuliert, den wir aber mit dem gleichen Wert abschätzen. Insgesamt haben wir also bislang (wieder global betrachtet) einen Fehler von

$$\text{Fehler}(C) \leq 8\text{vol}(C)\ln\left(16\text{vol}(G)\right)\ln\ln\left(8\text{vol}(G)\right)$$

produziert, dafür ist unser Problem zumindest für C gelöst.

Äquivalent können wir den Rest, das Lösen des Problems auf dem Restgraphen $G - C$, angehen, wobei dann höchstens der gleiche Fehlerterm wie oben entsteht (mit C ersetzt durch $G - C$). Setzen wir beide Terme zusammen, erhalten wir den totalen Fehler, der insgesamt bei allen Divide and Conquer-Schritten der Rekursion anfällt.

$$\text{Fehler}(G) \leq 8\text{vol}(G)\ln\left(16\text{vol}(G)\right)\ln\ln\left(8\text{vol}(G)\right)$$

Wir wissen also, daß unsere per Divide and Conquer erhaltene Lösung um maximal diesen Fehler von der optimalen Lösung abweicht. Wir haben eine Approximation gefunden – sie sieht nur noch etwas seltsam aus:

$$\text{cost}_{\text{dq}}(G) \leq \text{cost}^*(G) + 8\text{vol}(G)\ln\left(16\text{vol}(G)\right)\ln\ln\left(8\text{vol}(G)\right)$$

Wir haben aber vorausgesetzt, daß das Volumen der Spreizmetrik auf unserem Graphen eine untere Schranke für die Kosten einer optimalen Lösung des Problems ist, d.h. $\text{vol}(G) \leq \text{cost}^*(G)$. Wir schätzen daher weiter ab,

$$\begin{aligned}\text{cost}_{\text{dq}}(G) &\leq \text{cost}^*(G) + 8\text{cost}^*(G)\ln\left(16\text{cost}^*(G)\right)\ln\ln\left(8\text{cost}^*(G)\right)\\ &= \text{cost}^*(G)\cdot O(\log(\text{cost}^*(G))\log\log(\text{cost}^*(G)))\end{aligned}$$

Damit ist gezeigt:

Satz 2 Der vorgestellte Divide and Conquer-Algorithmus approximiert die optimale Lösung $\tau = \mathrm{cost}^*(G)$ bis auf den Faktor $O(\log\tau\log\log\tau)$.

Bemerkung: Wird die Voraussetzng, daß zwei Kanten mit „großem“ Abstand $\mathrm{dist}(u,v) \geq |V|-1$ existieren, um einen konstanten Faktor $\alpha < 1$ verfehlt, d.h. es existieren stets Kanten mit $\mathrm{dist}(u,v) \geq \alpha(|V|-1)$, so funktioniert der Algorithmus trotzdem, die Abschätzung verschlechtert sich aber um den (konstanten) Faktor $\frac{1}{\alpha}$.

Noch eine Bemerkung: Es ist recht egal, ob im logarithmischen Term des Approximationsfaktors die Knotenanzahl, Kantenanzahl, oder die „optimalen Kosten“ τ genannt werden, da $|E| = O(|V|^2)$ und (z.B. für Linear Arrangement) $\tau = O(|V|^3)$ und sich deshalb die logarithmierten Werte nur um einen konstanten Faktor unterscheiden.

5 Bestimmung der Spreizmetrik

Der Algorithmus verläßt sich darauf, daß eine Spreizmetrik mit den gewünschten Randbedingungen für jede Probleminstanz effizient berechnet werden kann. Die Spreizmetrik kann meist über ein lineares Programm berechnet werden, so z.B. für Linear Arrangement auf dem Graphen $G = (V,E)$:

$$\begin{array}{ll} \min & \sum_{e\in E} l(e) \\ \text{s.t.} & \forall U \subseteq V, \forall v \in U : \sum_{u\in U} \mathrm{dist}(u,v) \geq \frac{1}{4}(|U|^2-1) \\ & \forall e \in E : l(e) \geq 0 \end{array}$$

Das lineare Programm hat zwar exponentiell viele Nebenbedingungen, da aber eine verletzte Bedingung in polynominaler Zeit gefunden werden kann, ist eine Lösung z.B. mit dem Ellipsoidalgorithmus möglich.

Man bemerke, daß die Forderung an die Summe von Distanzen keine lineare Nebenbedingung ist, und obiges Programm erst zum linaren Programm wird, wenn die Distanzen für indirekte Wege als Hilfsvariablen eingeführt werden und diese durch die Dreiecksungleichung begrenzt werden. Eine verletzte Nebenbedingung kann dann mit einem an jedem Punkt gestarteten „single source shortest path“-Algorithmus gefunden werden, wobei die entsprechende Bedingung in jedem Schritt getestet wird.

Es kann gezeigt werden, daß auch in einem optimalen Linear Arrangement die Bedingung $\sum_{u\in U} \mathrm{dist}(u,v) \geq \frac{1}{4}(|U|^2-1)$ erfüllt ist, so daß eine Lösung des linearen Programms – mit den gleichen Nebenbedingungen wie das Original, aber mit viel größerem Lösungsraum – auch immer untere Schranke für die optimale Lösung ist.

Auch die letzte Bedingung an unsere Spreizmetrik, die Existenz zweier Knoten mit Abstand $\mathrm{dist}(u,v) \geq \alpha(|U|-1)$ in jedem Teilgraphen $U \subseteq V$, ist für $\alpha = \frac{1}{4}$ erfüllt: Aufgrund unserer Bedingung an die Summe der Abstände ist in jedem Teilgraphen der mittlere Abstand eines Knotens von u $\frac{1}{4}(|U|+1)$, und es existiert mindestens ein Knoten, dessen Abstand von u größer oder gleich dem Durchschnitt ist.