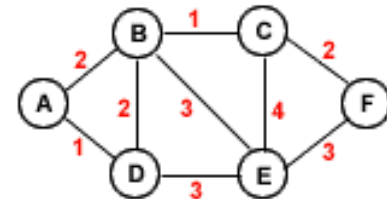




Effiziente Algorithmen

Hartmut Klauck
Universität Frankfurt
SS 06
20.7.





Set Cover

- Ein ähnliches Ergebnis wie bei der greedy Approximation erhält man durch Formulierung als ein LIP, und eine folgende randomisierte Rundung
- Seien S_1, \dots, S_m die Elemente von F
- Verwende Variablen z_1, \dots, z_m
- Min $\sum z_i$
 - Bed.: Für alle x aus U : $\sum_{i: x \in S_i} z_i \geq 1$
 - z_i aus $\{0, 1\}$
- Relaxiere zu einem LP und löse

Randomisierte Rundung

- Wir runden die erhaltene Lösung so:
 - z_i ist eine Zahl aus dem Intervall $[0,1]$. Mit Wahrscheinlichkeit z_i nehmen wir S_i auf
- Erwartete Kosten entsprechen dem Optimum des LP
- Allerdings überdecken wir wahrscheinlich nicht das ganze Universum
- Für jedes Element $x \in U$ gilt: $\sum_{i: x \in S_i} z_i \geq 1$
- Die Wahrscheinlichkeit, x nicht zu überdecken ist $\prod_{i: x \in S_i} (1-z_i)$
- Es gilt $\prod_{i: x \in S_i} (1-z_i) \leq (1-1/l)^l \leq 1/e$ wenn x in l Mengen S_i ist:
arithmetisches vs. geometrisches Mittel
- Damit werden erwartet $|U|/e$ Elemente nicht überdeckt.
- Wir iterieren dies erwartet $\ln |U|$ mal und erhalten eine Lösung mit Verlustfaktor $\ln |U|$



Traveling Salesman

- Im Traveling Salesman Problem ist eine Matrix von Distanzen zwischen n Städten
- Gesuchte wird eine Permutation der n Städte (Rundreise), so daß $\sum_{i=1 \dots n+1} \text{dist}(\pi(i), \pi(i+1 \bmod n))$ minimal ist
- Problem ist NP-schwer
- Bei maximaler Gewichtsgröße W ist eine W/n Approximation NP-schwer
- W ist i.A. exponentiell in der Eingabelänge (wenn Distanzen binär kodiert sind)



Traveling Salesman

- Zusätzliche Struktur:
 - Dreiecksungleichung
 - Euklidisch, d.h. die Distanzen sind Distanzen zwischen n Punkten in einem euklidischen Raum
- In beiden Fällen bleibt das Problem NP-schwer
- Im euklidischen Fall gibt es ein Approximationsschema!



Traveling Salesman mit Δ -Ungleichung

- Distanzen folgen der Dreiecksungleichung, d.h. $\text{dist}(u,v) + \text{dist}(v,w) \geq \text{dist}(u,w)$
- Wir folgen dem Ansatz, eine ähnliche, aber einfachere Struktur in polynomieller Zeit zu bestimmen
- Berechne einen minimalen Spannbaum
- Durchlaufe die Kanten des Spannbaums zweimal, um eine Tour zu erhalten, die alle Knoten mindestens einmal durchläuft
- Füge jetzt „Abkürzungen“ ein, um eine Tour zu erhalten, die jeden Knoten genau einmal besucht
- Die neue Tour ist nicht länger als die alte wegen der Dreiecksungleichung
- Da eine Traveling Salesman Tour immer mindestens so teuer ist wie ein minimaler Spannbaum erhalten wir so eine Salesman Tour, die eine 2-Approximation ist
- Besser: 3/2 Approximation (Christofides)



Max Cut

- Wir wollen nun einen besseren Approximationsalgorithmus für Max Cut entwerfen
- Wir folgen dem Ansatz
 - Integer Programm
 - Relaxierung
 - randomisierte Rundung
- [Goemans Williamson 94]



Max Cut

- Gegeben sei ein gewichteter ungerichteter Graph
- Wir suchen eine Partition der Knoten in zwei Mengen, so daß die Menge kreuzender Kanten maximales Gewicht hat
- Wir kennen schon eine 2-Approximation: wähle die Partition zufällig (oder greedy)



Ein Integer Programm

- Variablen y_1, \dots, y_n
- Gewichte des Graphen seien $w(i, j)$
- $\text{Max } \frac{1}{2} \sum_{i < j} w(i, j)(1 - y_i y_j)$
 - Bedingung: y_i aus $\{-1, 1\}$ für alle i
- Quadratisches Integer Programm
 - NP schwer zu lösen
- Klar: Optimum ist = Max Cut



Eine Relaxation

- Wir betrachten y_i als einen 1-dimensionalen Einheitsvektor
- Wir wechseln in einen n -dimensionalen Raum
- erlauben weitere Einheitsvektoren
- Relaxation bedeutet, daß eine optimale Max Cut Lösung (bestehend aus 1-dim Vektoren) zulässig ist und den selben Wert der Zielfunktion besitzt
- Wir verwenden also Variablen v_i aus \mathbb{R}^n , die Einheitsvektoren sein sollen
- Produkt der v_i ?
 - Skalarprodukt!



Eine Relaxation

- $\text{Max } \frac{1}{2} \sum_{i < j} w(i, j)(1 - v_i \cdot v_j)$
 - Bedingung: v_i ist Einheitsvektor in \mathbb{R}^n für alle i
 - d.h. v_i aus S_n , der n -dim Sphäre
- Produkt der v_i ist Skalarprodukt
- Klar: wenn alle v_i aus einem 1-dim Teilraum kommen, erhalten wir eine „normale“ Relaxation
- Wir müssen noch zeigen, wie diese Relaxation zu lösen ist!



Der Max Cut Algorithmus

1. Löse das relaxierte Programm
 2. Wir erhalten Vektoren v_1, \dots, v_n aus S_n
 3. Wähle einen Vektor r uniform zufällig aus S_n
 4. Sei $L = \{i: v_i \cdot r \geq 0\}$
 5. $L, V-L$ ist die Ausgabe
- - Geometrisch betrachtet wählen wir eine zufällige Hyperebene durch den Ursprung
 - Die Vektoren bzw. ihre Knoten werden aufgeteilt in die beiden Halbräume
 - Es besteht die Tendenz, Knotenpaare mit viel Kantengewicht zu trennen
 - Das relaxierte Programm ist unabhängig vom Koordinatensystem

Die Approximationsqualität

- Sei $W=(L,R)$ der produzierte Schnitt
- $E[W]$ ist das erwartete Gewicht der kreuzenden Kanten
- Wir zeigen:
- **Theorem 25.1:**
$$E[W] = \sum_{i < j} w(i,j) \arccos(v_i \cdot v_j) / \pi$$
- Sowie
- **Theorem 25.2:**
$$E[W] \geq \alpha \frac{1}{2} \sum_{i < j} w(i,j) (1 - v_i \cdot v_j)$$
 - für $\alpha = \min_{0 \leq \theta \leq \pi} 2/\pi \cdot \theta / (1 - \cos \theta)$



Die Approximationsqualität

- Wenn z^* die optimale Lösung des relaxierten Programms ist, und opt die optimale Lösung des Max Cut Problems, dann gilt $z^* \geq \text{opt}$
- $z^* = \text{Max} \frac{1}{2} \sum_{i < j} w(i, j)(1 - v_i \cdot v_j)$
- $E[W] \geq \alpha \text{Max} \frac{1}{2} \sum_{i < j} w(i, j)(1 - v_i \cdot v_j) \geq \alpha \text{opt}$
- Man kann ausrechnen, daß $\alpha \geq 0.878$
- Wir erhalten also eine 1.139-Approximation



Theorem 25.1

- **Theorem 25.1:**

$$E[W] = \sum_{i < j} w(i, j) \arccos(v_i \cdot v_j) / \pi$$

- **Beweis:**

- Wir haben Vektoren v_1, \dots, v_n

- $E[W]$ ist die erwartete Schnittgröße, wenn wir r aus S_n zufällig wählen und entsprechend aufteilen

- $E[W] = \sum_{i < j} w(i, j) \Pr[\text{sgn}(v_i \cdot r) \neq \text{sgn}(v_j \cdot r)]$
per Linearität des Erwartungswertes

- $\text{sgn}(x) = 1$ wenn $x \geq 0$ und -1 sonst

- **Lemma:**

$$\Pr[\text{sgn}(v_i \cdot r) \neq \text{sgn}(v_j \cdot r)] = 1/\pi \cdot \arccos(v_i \cdot v_j)$$

- Mit dem Lemma folgt das Theorem direkt.



Das Lemma

- Gegeben sind Vektoren u und v aus S_n
- Wir ziehen r zufällig
- Ws, daß eine zufällige Hyperebene u und v trennt?
- Entspricht dem Winkel zwischen u und v !

- Der Winkel ist $\theta = \arccos(uv)$
- Per Symmetrie gilt $\Pr[\text{sgn}(ur) \neq \text{sgn}(vr)] = 2\Pr[ur \geq 0 \text{ und } vr < 0]$
- Die Menge $\{r: ur \geq 0 \text{ und } vr < 0\}$ ist der Schnitt zweier Halbräume mit Winkel θ
- Der Schnitt mit S_n hat Größe $\theta/2\pi$
- Damit folgt $\Pr[ur \geq 0 \text{ und } vr < 0] = 1/2\pi \cdot \arccos(v_i v_j)$
- Das Lemma folgt



Theorem 25.2

- **Theorem 25.2:**
 $E[W] \geq \alpha \frac{1}{2} \sum_{i < j} w(i, j) (1 - v_i \cdot v_j)$
 - für $\alpha = \min_{0 \leq \theta \leq \pi} 2/\pi \cdot \theta / (1 - \cos \theta)$
- Intuition: Wir schätzen den arccos linear ab
- **Lemma:**
Für $-1 \leq y \leq 1$: $\arccos(y) / \pi \geq \alpha \frac{1}{2} (1 - y)$
- Damit folgt das Theorem, denn
 $E[W] = \sum_{i < j} w(i, j) \arccos(v_i \cdot v_j) / \pi$



Lemma

- **Lemma:**
Für $-1 \leq y \leq 1$:
 $\arccos(y) / \pi \geq \alpha \frac{1}{2} (1-y)$
für $\alpha = \min_{0 \leq \theta \leq \pi} 2/\pi \cdot \theta / (1 - \cos \theta)$

- **Beweis:** setze $\cos \theta = y$



Das relaxierte Programm

- $\text{Max } \frac{1}{2} \sum_{i < j} w(i, j)(1 - v_i \cdot v_j)$
 - Bedingung: v_i ist Einheitsvektor in \mathbb{R}^n für alle i
 - d.h. v_i aus S_n , der n -dim Sphäre
- **Fakt:**
 - Eine symmetrische Matrix A ist positiv semidefinit gdw es eine Matrix B gibt mit $A = B^T B$
 - Dabei kann B eine $m \times n$ Matrix sein mit $m \leq n$
 - Eine Matrix B mit vollem Rang kann in Zeit $O(n^3)$ berechnet werden durch Cholesky Dekomposition



Das relaxierte Programm

- $Y=B^T B$ sei positiv semidefinit und habe $y_{i,i}=1$
- Damit entspricht Y einer Menge von Einheitsvektoren v_1, \dots, v_n aus S_m
- v_i ist i -te Spalte von B
- $Y(i,j)=v_i \cdot v_j$
- Y heißt die Gram Matrix von v_1, \dots, v_n
- Wir formulieren ein neues Programm:
 - Max $\frac{1}{2} \sum_{i < j} w(i,j) (1-y(ij))$
 - Bedingung: $y(i,i)=1$ und y pos. semidefinit
- Wir erhalten ein semidefinites Programm, das in polynomieller Zeit gelöst werden kann (bis auf beliebig kleines ε)
- Wir erhalten die Matrix y und können die v_i per Cholesky berechnen



Zusammenfassung

- Das Max Cut Problem kann in polynomieller Zeit mit Faktor 1.139 approximiert werden
- Statt der einfachen 2-Approximation
- Die bestmögliche Approximation in polynomieller Zeit ist schlechter als 1.0624, außer „NP ist einfach“
- Unter Zusatzannahmen ist GW Algorithmus sogar optimal