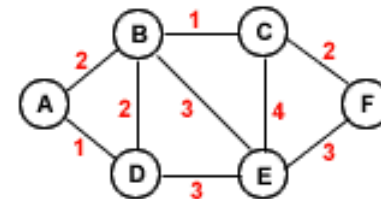




Effiziente Algorithmen

Hartmut Klauck
Universität Frankfurt
SS 06
18.7.





Approximation durch Zufall

- Manche Optimierungsprobleme können gut durch zufällige Wahl der Lösung approximiert werden
- Oft bei Constraintproblemen
- Beispiel: Max 3-SAT
 - Gegeben eine Formel in 3-KNF
 - n Boolesche Variablen und m Terme $L_{i,1} \vee L_{i,2} \vee L_{i,3}$, wobei die L Variablen oder negierte Variablen sind
 - Gesucht wird eine Lösung, die die maximale Anzahl von Termen erfüllt
- Max 3-SAT ist NP schwer, da 3-SAT NP vollständig



Approximation durch Zufall

- Approximative Lösung für Max 3-SAT:
 - Ziehe alle n Variablen zufällig
- Jeder Term wird mit Wahrscheinlichkeit $1 - 1/8 = 7/8$ erfüllt
- Die erwartete Anzahl erfüllter Terme ist somit:
 - Sei z_i eine Indikatorzufallsvariable, die 1 ist wenn Term i erfüllt ist
 - Erwartete Anzahl erfüllter Terme ist $E[\sum z_i] = \sum E[z_i] = m \cdot 7/8$
- Wir erhalten somit erwartet eine $7/8$ Approximation
- Bei einem randomisierten Algorithmus wollen wir allerdings normalerweise eine gute Lösung mit hoher Wahrscheinlichkeit (z.B. $2/3$), nicht nur erwartet



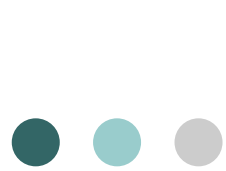
Approximation durch Zufall

- Ähnlicher Ansatz für viele Probleme der Form „maximiere Anzahl erfüllter Constraints“
- Auch Max Cut
- Für Max 3-Sat ist diese Lösung sogar optimal, außer NP-vollständige Probleme sind „einfach“



Methode des bedingten Erwartungswerts

- Wir wollen den obigen Algorithmus derandomisieren
- Betrachte die n Zufallswahlen als einen Baum, dessen Blätter mit den strings der Länge n markiert sind
- Der Erwartungswert $E_x[\sum_i z_i(x)]$ geht über alle Blätter x_1, \dots, x_n
- Wir fixieren x_1 usw. nacheinander greedy.
- Dazu betrachten wir die bedingten Erwartungswerte $E_x[\sum z_i(x)|x_1=1]$ und $E_x[\sum z_i(x)|x_1=0]$
- Da $7/8 \cdot m \leq E_x[\sum_i z_i(x)] = 1/2(E_x[\sum z_i(x)|x_1=1] + E_x[\sum z_i(x)|x_1=0])$ muss einer von beiden größer als $7/8 m$ sein
- Fixiere x_1 so, daß der Erwartungswert über die restlichen Variablen maximal ist und iteriere.
- Am Ende erhalten wir eine Lösung mit Zielfunktion mind. $7/8 m$
- Allerdings: wie können wir die bedingten Erwartungswerte deterministisch bestimmen?
- Wir wissen, daß für alle Formeln mit m Klauseln die $7/8 m$ Schranke gilt, daher können wir x_1 so setzen, daß möglichst viele Klauseln direkt erfüllt werden, und wir erhalten insgesamt eine Festlegung von x_1 , so daß der bedingte Erwartungswert mind. $7/8 m$ ist
- Wir erhalten den greedy $8/7$ Approximationsalgorithmus:
 1. Von $i=1$ bis n : gibt es mehr Klauseln mit x_i als mit $\neg x_i$ so setze $x_i=1$, sonst $x_i=0$



VC und lineare Programmierung

- Wir suchen ein minimales VC in Graphen mit Knotengewichten
- Wir beginnen mit einem LIP:
 - Variablen $x(v)$ für die Knoten
 - $x(v)=1$ gdw v im VC liegt
 - $w(v)$ sei Gewicht des Knotens
 - Minimiere $\sum_v w(v) x(v)$
 - Bedingungen:
 - $x(u)+x(v) \geq 1$ für alle Kanten
 - $x(v) \in \{0,1\}$ für alle Knoten



Eine Relaxation

- Wir bestimmen nun eine Relaxation des LIP in ein LP:
 - $\text{Min } \sum w(v), x(v)$
 - Bedingungen:
 - $x(u)+x(v) \geq 1$ für alle Kanten (u,v)
 - $x(v) \leq 1$ und $x(v) \geq 0$ für alle Knoten
- Das LP kann mit Ellipsoid oder Interior Point in polynomieller Zeit gelöst werden, oder mit Simplex in der Praxis
- Problem: $x(v)$ sind nicht ganzzahlig



Runden der Lösung

- Wir durchlaufen die Lösung $x(1)..x(n)$
- Wenn $x(i) \geq \frac{1}{2}$, so legen wir i in die Menge C
- C wird ausgegeben als VC

- Behauptung: Wir erhalten einen 2-Approximationsalgorithmus mit polynomieller Laufzeit

- Laufzeit: Klar



Korrektheit

- Sei C^* ein optimales VC
- Sei z^* der optimale Wert des LP
- C^* ist eine Lösung des LP, daher gilt:
 - $z^* \leq w(C^*)$
- Noch zu zeigen: C ist ein VC und $w(C)$ ist 2-Approximation
- Erstens: Für alle Kanten gilt $x(u)+x(v) \geq 1$, also muß $x(u) \geq 1/2$ oder $x(v) \geq \frac{1}{2}$ gelten
- Damit liegt u oder v in C und C ist ein VC

Approximation

$$\begin{aligned} \circ z^* &= \sum_v x(v) \cdot w(v) \\ &\geq \sum_{v: x(v) \geq 1/2} x(v) \cdot w(v) \\ &\geq \sum_{v: x(v) \geq 1/2} w(v) / 2 \\ &= \sum_{v \in C} w(v) / 2 = \frac{1}{2} \cdot w(C) \\ \Rightarrow w(C) &\leq 2z^* \leq 2 \cdot w(C^*) \end{aligned}$$



Bemerkung

- Max weiss, daß VC kein Approximationsschema hat ausser NP ist „einfach“
- Unter weiteren Annahmen aus der Theorie der probabilistischen NP-Beweissysteme ist sogar eine $2-\varepsilon$ Approximation in polynomieller Laufzeit nicht möglich



Set Cover

- Verallgemeinerung von Vertex Cover
- Gegeben ein Universum $U = \{1, \dots, n\}$ sowie eine Familie F von Teilmengen von U so daß jedes Element von U in mindestens einer Teilmenge liegt
- Wir sagen eine Menge von Elementen m_1, \dots, m_l von F überdeckt U , wenn $\cup m_i = U$
- Gesucht wird eine minimale Anzahl von Elementen von F die U überdecken
- VC: Universum: Kanten, Teilmengen sind Knoten, d.h. Mengen der adjazenten Kanten
 - VC hat die Einschränkung, daß jedes Universumselement in genau zwei Teilmengen liegt



Set Cover

- Naheliegender Greedy Algorithmus:
 - Wähle die Teilmengen, welche die meisten Universumselemente überdeckt
 - Entferne die Teilmenge und die überdeckten Elemente
 - Iteriere bis alles überdeckt
- Die Laufzeit ist klar polynomiell und wir erhalten eine Überdeckung
- Wie gut ist die Approximation??



Set Cover

- Sei $H_d = \sum_{i=1, \dots, d} 1/i$
- $H_d \leq \ln d + 1$
- **Theorem:**
 - Sei m die maximale Mengengröße eines Elementes von F
 - Der greedy Set Cover Algorithmus erzielt eine H_m Approximation
 - Damit ist Set Cover $\ln n$ -approximierbar in polynomieller Zeit
- Bemerkung: Man kann zeigen, daß eine $(1-\varepsilon) \ln n$ Approximation nicht in polynomieller Zeit möglich ist, außer NP ist „einfach“



Set Cover

- Sei C^* ein optimales Set Cover und C das im greedy Algorithmus erreichte
- S_i sei die im i -ten Schritt gewählte Menge
- x seien Elemente von U
- Die Kosten eines Schrittes sind 1, wir teilen diese Kosten den überdeckten Elementen zu
- Ein Element von U erhält Kosten nur dann, wenn es zuerst überdeckt wird
- c_x seien Kosten von x , x werde in Schritt i überdeckt, dann sind Kosten von x :

- $c_x = \frac{1}{|S_i - S_1 \cup \dots \cup S_{i-1}|}$



Set Cover

- Es gilt $|C| = \sum_x c_x$
- Betrachte nun $\sum_{S \in C^*} \sum_{x \in S} c_x \geq \sum_{x \in U} c_x = |C|$
- Wir zeigen nun, daß $\sum_{x \in S} c_x \leq H_{|S|}$
- Damit gilt das Theorem über die Approximationsqualität, denn
 - $|C| \leq |C^*| \cdot H_d$, wobei d die maximale Mengengröße ist.



Set Cover

- Betrachten wir also eine Menge S aus F
- Wir sind an $\sum_{x \in S} c_x$ interessiert
- Sei $S = \{a_1, \dots, a_d\}$, angeordnet nach dem Zeitpunkt, zu dem die Elemente jeweils zum ersten Mal überdeckt wurden
- Sei k der letzte Schritt in dem ein Element von S (genauer gesagt a_d) zum ersten Mal überdeckt wird, mit Kosten $c(a_d) \leq 1$
- Als a_{d-1} (eventuell vor Schritt k) überdeckt wurde, wurde noch mindestens ein zweites Element von U überdeckt, denn sonst wäre S eine bessere Wahl gewesen, also $c(a_{d-1}) \leq \frac{1}{2}$
- usw.
- Insgesamt gilt $\sum_{x \in S} c_x \leq (1 + 1/2 + 1/3 + \dots + 1/d) = H_{|S|}$