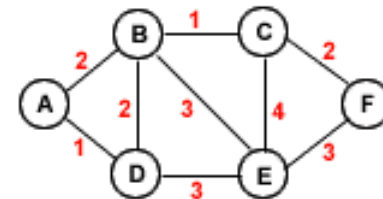


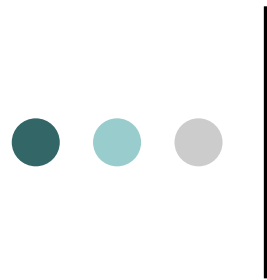


Effiziente Algorithmen

Hartmut Klauck
Universität Frankfurt
SS 06

13.7.





Lineare Integer Programme

- Wir erlauben Variablen, die nur ganzzahlig sein dürfen
- Sonst wie LP
- Man sieht leicht NP-Vollständigkeit

- Ansatz für NP-Optimierungsprobleme:
 - Formuliere als LIP
 - Relaxiere zu einem LP durch Streichen der Integer Constraints
 - Runde die erhaltenen Lösung



Semidefinite Programme

- Ausdrucksstärkere Erweiterung von LP
- Immer noch in polynomieller Zeit zu lösen
 - Ellipsoid/Interior Point Methoden
- Kein „Simplex“ Algorithmus bekannt



Begriffe

- Eine reelle symmetrische $n \times n$ Matrix heißt positiv semidefinit wenn
 - $v^T A v \geq 0$ für alle Vektoren v
- Insbesondere sind alle Eigenwerte reell und ≥ 0
- positiv definit: mit $>$
- Notation: $X \geq 0$: X ist pos. semidef.
- $X \geq Y$ wenn $X - Y$ pos. semidefinit
- **Fakt:** wenn A pos. semidef ist, dann gibt es orthogonale Transformation O , so daß $O A O^T$ diagonal ist und alle Einträge der Diagonalen ≥ 0 (Eigenwerte)



Semidefinites Programm

- Ähnlich zu linearem Programm,
- Wir denken uns die Variablen als Matrix angeordnet, mit dem Constraint, dass die Matrix pos. semidefinit ist.
- Variablenmatrix X
 - Minimiere $\sum_{i,j} c_{i,j} X[i,j]$
 - Constraints:
 - $\sum_{i,j} A_k[i,j] X[i,j] = b_k$ für $k=1,\dots,m$
 - A_i sind symmetrische Matrizen
 - $X \succeq 0$ (d.h. X ist pos. semidef)



Bemerkung

- Tatsächlich wird das Infimum gebildet, da ein Minimum nicht existieren muss



Fakten

- Das Optimum eines semidefiniten Programms kann in polynomieller Zeit gefunden werden
- Genauer gesagt kann das Infimum beliebig nah approximiert werden, um bis auf ε heranzukommen kostet Laufzeitfaktor $\log(1/\varepsilon)$
- Zu einem semidefiniten Programm gibt es ein duales Programm und es gilt schwache Dualität, d.h. zulässige Lösungen des primalen Programms haben immer eine größere Zielfunktion als zulässige Lösungen des dualen



Das Duale SDP

- Das duale SDP zu einem SDP C, A, b ist $\sup \sum b_i y_i$ unter der Bedingung $\sum y_i A_i \leq C$
- Schwache Dualität:
Zielfunktion des primalen ist \geq Zielfunktion des dualen für alle zul. Lösungen des primalen und des dualen
- Starke Dualität gilt nicht allgemein, aber in Spezialfällen



Eine Anwendung

- Sei G ein ungerichteter ungewichteter Graph
- Eine unabhängige Menge ist eine Menge von Knoten in der kein Paar durch eine Kante verbunden ist
- $\alpha(G)$ sei die Größe einer maximalen unabhängigen Menge
- Ein Graph ist k -färbbar, wenn die Knoten mit k Farben gefärbt werden können, so daß keine zwei adjazenten Knoten dieselbe Farbe haben
- Die chromatische Zahl $\chi(G)$ ist die minimale Anzahl von Farben um G zu färben
- $\omega(G)$ sei die Größe einer maximalen Clique
- Klar: $\omega(G) \leq \chi(G)$



Eine Anwendung

- Alle diese Parameter sind NP-schwer zu berechnen
- Wir definieren einen Parameter $\vartheta(G)$
- „Lovasz Theta Funktion“
- Sei P die Menge aller symmetrischen Matrizen A mit $A(i,j)=1$ wenn (i,j) keine Kante in G ist oder $i=j$
- Wenn eine unabhängige Menge der Größe k existiert haben alle Matrizen in P eine $k \times k$ Teilmatrix aus Einsen mit k Einträgen auf der Hauptdiagonalen
- Damit gilt, daß der größte Eigenwert $\lambda_{\max}(A)$ von jeder Matrix A in P mindestens k ist
- Setze $\vartheta(G) = \min_{A \in P} \lambda_{\max}(A) \geq \alpha(G)$
- Wir erhalten so eine obere Schranke für $\alpha(G)$
- [Analog $\omega(G) \leq \vartheta(\text{Kompl. } G)$]



Eine Anwendung

- Semidefinites Programm für $\vartheta(G)$:
 - Es gilt: $\lambda_{\max}(A) = \min\{t : tI - A \geq 0\}$
 - Denn die Eigenwerte von $tI - A$ sind $t - \lambda_i$ für die Eigenwerte λ_i von A
 - $A \in P$ gdw $A - J$ eine Linearkombination von Matrizen $E_{i,j}$ für Kanten (i,j) des Graphen ist
 - Dabei: J ist die Matrix mit nur Einsen
 - $E_{i,j}$ Matrix mit genau einer Eins an (i,j) und (j,i) und sonst Nullen
 - Das semidefinite Programm:
Variablen $t, x_{i,j}$
 $\vartheta(G) = \min t:$
 $tI - \sum_{(i,j) \text{ Kante}} x_{i,j} E_{i,j} \geq J$
- Dieses Programm läßt sich leicht in eine Standardform bringen:
Verwende neue Variablen $Y_{i,j}$. Matrix Y muss pos. semidef sein.
Gleichungen: $Y_{1,1} = Y_{2,2} = \dots = Y_{n,n}$ sowie $Y_{j,j} = -1$, wenn (i,j) keine Kante ist,
 $Y_{i,j}$ ohne Constraint, wenn (i,j) eine Kante ist. Zielfunktion $Y_{1,1} + 1$
- Somit läßt sich $\vartheta(G)$ in polynomieller Zeit berechnen



Eine Anwendung

- Es gilt ebenfalls: $\vartheta(\text{Kompl. } G) \leq \chi(G)$
- Damit ist ein Parameter zwischen $\omega(G)$ und $\chi(G)$ berechenbar in polynomieller Zeit
- Ein Graph heißt *perfekt*, wenn $\omega(H) = \chi(H)$ für alle induzierten Subgraphen H von G (ein induzierter Subgraph ist eine Teilmenge der Knoten zusammen mit allen Kanten zwischen diesen)
- **Korollar:** Für perfekte Graphen kann die Cliquengröße in polynomieller Zeit berechnet werden
- Wann ist ein Graph perfekt?
- **Perfect Graph Theorem:**
 - Ein Graph ist perfekt gdw er kein ungerades Loch oder Antiloche enthält
- Loch: Zyklus der Länge >3 ohne „Speichen“
- Antiloche: Komplement eines Loches
- Zum Beispiel ist ein Zyklus der Länge 5 ein ungerades Loch
- Insbesondere ist ein Loch ein induzierter Subgraph mit Cliquengröße 2, der nur 3-färbbar ist
- **Perfektheit kann in polynomieller Zeit entschieden werden**



NP Optimierungsprobleme

- Ein NP Optimierungsproblem hat Instanzen und Lösungen und ist gegeben durch eine Kostenfunktion, die in polynomieller Zeit berechnet werden kann und eine Zulässigkeitsfunktion, die in P liegt
- Es soll eine zulässige Lösung maximaler oder minimaler Kosten berechnet werden
- „NP“: Die Sprache „ist das Optimum $\geq t$ “ liegt in NP für ein NP Optimierungsproblem:
 - Rate Lösung, Teste Zulässigkeit
 - Berechne Kosten, Test ob $\geq t$
- Wir nennen ein NP Optimierungsproblem NP-schwer, wenn seine (oder irgendeine seiner) Sprachversion NP-vollständig ist



Approximation

- Ein Algorithmus berechne zu einem NP-Optimierungsproblem bei einer gegebenen Problem Instanz zulässige Lösungen x , welche erfüllen:
 - $\text{cost}(x) \leq \text{cost}(\text{opt}) \cdot c$ bei Minimierung
 - $\text{cost}(x) \geq \text{cost}(\text{opt})/c$ bei Maximierung
 - $c \geq 1$
- Dann sagen wir, daß der Algorithmus ein c -Approximationsalgorithmus ist bzw. Verlustfaktor c hat



Beispiel

- Max Cut ist NP-schwer, hat aber einen 2-Approximationsalgorithmus mit polynomieller Laufzeit



Begriffe

- Ein NP Optimierungsproblem ist konstant approximierbar, wenn es einen c -Algorithmus mit polynomieller Laufzeit gibt und $c=O(1)$
- Generalisierbar auf Funktionen $f(n)$
z.B. $O(\log n)$ -Approximation
- Ein NP Optimierungsproblem besitzt ein Approximationsschema (PTAS), wenn es für jede Konstante $\varepsilon > 0$ einen $1+\varepsilon$ Approximationsalgorithmus mit polynomieller Laufzeit gibt
 - Problem: Laufzeit kann $n^{1/\varepsilon}$ sein
- Ein NP-Optimierungsproblem hat ein volles Approximationsschema (FPTAS), wenn es ein PTAS hat und die Laufzeit von der Form $\text{poly}(n, 1/\varepsilon)$ ist



Vertex Cover

- Wir betrachten ungerichtete ungewichtete Graphen
- Ein Vertex Cover ist eine Menge W von Knoten, so daß jede Kante mindestens einen Endpunkt in W hat
- Gesucht ist ein minimal großes Vertex Cover
- Das Problem ist NP-schwer:
 - Das Komplement eines minimalen Vertex Cover ist eine maximale unabhängige Menge



Ein

Approximationsalgorithmus

- Greedy Methode
 - Starte mit leerer Menge C
 - Durchlaufe Kanten:
 - Wenn beide Endpunkte nicht in C liegen, füge beide in C ein
 - Sonst nicht
 - Gib C aus
- Laufzeit: $O(n+m)$
- Korrektheit: Die Ausgabe ist klar ein Vertex Cover
- Approximationsqualität:
 - Sei M die Menge der gewählten Kanten
 - M ist ein Matching
 - Jedes Vertex Cover C' muss mindestens einen der Endpunkte jeder Kante in M haben, C hat jeweils beide
 - Da die Kanten in M ein Matching sind, kann jede Knoten in C' nur eine Kante in M überdecken
 - Daher muss C' mindestens halb so groß wie C sein
- Bemerkung: obwohl die minimale VC Größe = $n - \alpha(G)$ können wir analog keine Approximation für $\alpha(G)$ erhalten