Universität Karlsruhe
Fakultät für Informatik
ITI Wagner

6. Musterlösung

Problem 1: Fingerabdrücke 2pt

Algorithmus 1 : MATRIZENPRODUKTTEST (A, B, C)

```
1 r ← ⟨ Vektor von n unabhängigen Zufallsbits ⟩
2 x ← Br
3 y ← Ax
4 z ← Cr
5 Wenn y ≠ z
6 │ return NO
7 sonst
8 │ return YES
```

(a) Der Algorithmus überprüft ob $A(Br) = Cr$ für einen zufällig gewählten Vektor r gilt. Wenn $AB = C$ gilt, dann gilt $A(Br) = Cr$ auch und der Algorithmus 1 liefert in diesem Fall YES.

(b) Sei $D = AB - C$, also $D \neq 0$. Außerdem setzen wir $y = ABr$ und $z = Cr$. Dann gilt, dass der Algorithmus 1 genau dann YES liefert, wenn $Dr = 0$. Es bezeichne d die erste Zeile von D. O.B.d.A sei d nicht der Nullvektor und die Nichtnulleinträge von d seinen genau die erste ℓ Einträge d_1,...,d_ℓ. Der erste Eintrag im Produkvektor Dr ist gerade dr. Wir suchen eine obere Schranke für die Wahrscheinlichkeit, dass $dr = 0$ ist. Das ist genau der Fall, wenn $\sum_{i=1}^{\ell} d_i r_i = 0$, also genau dannn, wenn $r_\ell = (-\sum_{i=1}^{\ell-1} d_i r_i)/d_\ell$. Um einzusehen, wie unwahrscheinlich dies ist, stelle man sich vor, r_1,...,$r_{\ell-1}$ seinen bereits gewählt. Dann kann offensichtlich für höchsten einen der beiden möglichen Werte 0 und 1 für r_ℓ obige Gleichung richtig sein. Also ist diese Wahrscheinlichkeit höchsten 1/2.

(c) Durch k-fache Wiederholung des Algorithmus kann die Fehlerwahrscheinlichkeit wie üblich auf 2^{-k} gedrückt werden.

Problem 2: Approximation der Größe einer Clique 3pt

Eine Clique ist ein vollständig verbundener Subgraph eines Graphen. Sei $G = (V, E)$ ein ungerichteter, zusammenhängender, einfacher Graph. Für jedes $k \geq 1$, $k \in \mathbb{N}$ sei $G^{(k)}$ der ungerichtete Graph $(V^{(k)}, E^{(k)}$, wobei $V^{(k)}$ die Menge aller geordneten k-Tupel von Knoten aus v ist und $E^{(k)}$ so definiert ist, dass $(v_1, v_2, \ldots, v_k)$ genau dann mit $(w_1, w_2, \ldots, w_k)$ benachbart ist, wenn für alle i mit $1 \leq i \leq k$ entweder der Knoten v_i in G mit w_i benachbart ist oder $v_i = w_i$ gilt.

(a) Beweisen Sie, dass die Größe der maximalen Clique in $G^{(k)}$ gleich der k-ten Potenz der maximalen Clique in G ist.

(b) Zeigen Sie, dass für das Problem, eine maximale Clique zu berechnen, ein Approximationschema mit vollständig polynomieller Laufzeit gibt, falls es einen Approximationsalgorithmus mit konstantem Approximationsverhältnis für dieses Problem gibt.

Lösung:

(a) Projektion des Problems
Seien c und $\widehat{c}$ die Kardinalitäten von Cliquen von maximaler Kardinalität in G respektive $G^{(k)}$. Zu zeigen ist $\widehat{c} = c^k$. Wir zeigen zunächst $\widehat{c} \geq c^k$, und dann $\widehat{c} \leq c^k$.

Sei $C := \{v_1, \ldots, v_c\}$ eine maximale Clique (i.e. von maximaler Kardinalität) in $G = (V, E)$ mit Kardinalität c. Betrachte die folgende Menge von Tupeln $C^{(k)} := \{(x_1, \ldots, x_k) \in V^k \mid x_i \in C \text{ für alle } i = 1, \ldots, k\}$. Offenbar gilt $C^{(k)} \subseteq V^{(k)}$, weiterhin ist die Bedingung für Adjazenz in $G^{(k)}$ für jedes Paar $v_i^{(k)}, v_i^{(k)} \in C^{(k)}$ erfüllt. Somit bildet $C^{(k)}$ in $G^{(k)}$ eine Clique der Kardinalität c^k. Dies beweist $\widehat{c} \geq c^k$.

Sei D eine Clique in $G^{(k)}$ mit den Elementen $d^1, \ldots, d^\ell$, dabei bezeichne $d^i = (d_1^i, \ldots, d_k^i)$ mit $d_j^i \in V$. Für $j = 1, \ldots, k$ sei der *Koordinatensack* $S_j = \{d_j^i | 1 \leq i \leq \ell\}$ die Menge aller Knoten aus V, die in der j-ten Koordinate der Elemente von D vorkommen. Da D Clique ist, ist jedes S_j Clique in G. Somit gilt $|D| \leq \prod_{j=1}^k |S_j|$. Sei nun $\widehat{D}$ eine maximale Clique (i.e. von maximaler Kardinalität) in $G^{(k)}$, und seien die zugehörigen Koordinatensäcke $\widehat{S}_1, \ldots, \widehat{S}_{\widehat{c}}$. Dann gilt wieder $|\widehat{D}| \leq \prod_{j=1}^k |\widehat{S}_j|$, wobei S_j die Koordinatensäcke von D sind. Da die Koordinatensäcke Cliquen in G sind, gilt $\widehat{S_j} \leq c$ für $1 \leq j \leq k$, und somit $|\widehat{D}| = \widehat{c} \leq c^k$.

Somit gilt Gleichheit. Darüberhinaus gilt für die Koordinatensäcke: $\widehat{c} = c^k \Rightarrow \widehat{S}_1 =, \ldots, \widehat{S}_k = C$ für eine maximale Clique C in G.

(b) "Konstantes Approximationsverhältnis ⇒ PAS?"
Angenommen es gäbe einen solchen Algorithmus $\mathcal{A}$ mit relativem Approximationsverhältnis $a \geq 1$. Sei C eine maximale Clique in G, und für ein gegebens k sei $\mathrm{OPT}^{(k)}$ eine maximale Clique in $G^{(k)}$. Nach (a) gilt dann $|C|^k = |\mathrm{OPT}^{(k)}|$. Man definiere das Schema wie folgt:

- Definiere k wie folgt:
$$k := \left\lceil \frac{\log a}{\log(1+\varepsilon)} \right\rceil$$
- Löse das Problem für $G^{(k)}$. Dann liefert $\mathcal{A}$ eine Clique $\overline{D}$ der Kardinalität $\bar{c}$. Dabei gilt: $\frac{|\mathrm{OPT}^{(k)}|}{|\overline{D}|} \leq a$.
- Sei $\overline{S}_i$ der Koordinatensack maximaler Kardinalität von $\overline{D}$. Dann sei D die Clique in $G^{(k)}$ mit Koordinatensäcken $S_j = \overline{S}_i, 1 \leq j \leq k$ (siehe (a)). Dann gilt $|D| \geq |\overline{D}|$ und somit auch $\frac{|\mathrm{OPT}^{(k)}|}{|D|} \leq a$.
- In G sei nun C' die nach (a) zu der Clique D in $G^{(k)}$ korrespondierende Clique. Dann gilt:
$$\frac{|C|}{|C'|} = \frac{\sqrt[k]{|\mathrm{OPT}^{(k)}|}}{\sqrt[k]{|D|}} \leq \sqrt[k]{a} = a^{\frac{1}{k}} \leq a^{\frac{\log(1+\varepsilon)}{\log a}} = a^{\log_a(1+\varepsilon)} = 1 + \varepsilon \ .$$
Somit gilt für die die Größe der approximiert maximalen Clique C in G die gewünschte relative Approximationsgüte $1 + \varepsilon$.

Die Konstruktion des Graphen $G^{(k)}$ mit n^k Knoten und $O(n^{2k})$ Kanten erfolgt in $O(n^{2k})$, und der Approximationsalgorithmus wird in Polynomialzeit auf einem Graphen mit n^k Knoten ausgeführt. Da auch noch der Rückschluss auf die Lösung in G unmittelbar erfolgt, läuft das gesamte Schema in Polynomialzeit.

Um nun zu zeigen, dass das obige PAS ein FPAS ist, müsste man noch zeigen dass die Laufzeit polynomiell in $1/\varepsilon$ ist. Dazu muss man also zeigen, dass n^k polynomiell in $1/\varepsilon$ ist. Da sich $n^{\left\lceil \frac{\log a}{\log(1+\varepsilon)} \right\rceil}$ superpolynomiell in $1/\varepsilon$ verhält, ist obige Konstruktion kein FPAS. Dies schließt natürlich noch nicht aus, dass die Existenz eines FPAS auf andere Weise gefolgert werden kann (ich halte dies aber für unwahrscheinlich).

Problem 3: Vertex-Cover 2pt

(a) Der Algorithmus liefert ein Vertex Cover, da für jede Kante, die nicht überdeckt ist, beide Endknoten zur Lösung hinzugenommen werden. Betrachten wir nun einen weiteren Algorithmus (Algorithmus 2), der bereits eine optimale Lösung C^* als Eingabe bekommt, und diese auch wieder ausgibt.

Algorithmus 2 : Exakter Algorithmus für Vertex Cover

Eingabe : Graph $G = (V, E)$, Optimallösung C^*
Ausgabe : C (Minimales Vertex Cover von G)

```
1 C ← ∅
2 für alle {v_i, w_i} ∈ E tue
3     Wenn {v_i, w_i} nicht durch C überdeckt ist
4         Wenn v_i ∈ C*
5             C ← v_i
6         Wenn w_i ∈ C*
7             C ← w_i
8 return C
```

Offensichtlich fügt der Approximationsalgorithmus höchstens doppelt so viele Knoten zu C hinzu wie Algorithmus 2, denn mindestens eine der beiden inneren Wenn–Abfragen in Zeile 3 bzw. 5 muss erfüllt sein, weil C^* schließlich eine Lösung ist, und somit zu jeder Kante mindestens einen inzidenten Knoten beinhalten muss. Es lässt sich leicht ein Beispiel konstruieren, das zeigt, dass diese Grenze scharf ist. Siehe dazu Abbildung 1. Somit liefert der Algorithmus eine Faktor–2–Approximation für das Problem Vertex Cover.

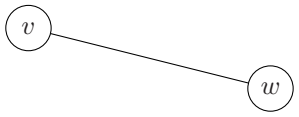

Abbildung 1: Ein optimales Vertex Cover würde nur einen der Knoten v oder w beinhalten. Der Algorithmus aus Aufgabe 3 (a) wählt jedoch beide Knoten aus.

(b) Betrachte den Graph in Abbildung 2. Eine gültige Folge, in der die Knoten im Algorithmus betrachtet werden können, ist die durch die Knotebeschriftung induzierte. Offensichtlich ist $|C| = 11$. Ein optimales Vertex Cover C^* ist durch die grauen Knoten gegeben, und hat eine Mächtigkeit von $|C^*| = 5$. Es gilt also $|C| > 2|C^*|$, und damit ist der Algorithmus kein Faktor–2–Approximationsalgorithmus für Vertex Cover.

(c) Sei $G = (V, E)$ ein Baum. Knoten $v \in V$ mit Grad 1 wollen wir als *Blatt* bezeichnen. Offensichtlich hat jeder Baum mit wenigstens einer Kante mindestens ein Blatt, da der Graph sonst nicht kreisfrei wäre.

Ist nun v ein beliebiger Blattknoten, so ist wegen $\text{Grad}(v) = 1$ das Blatt zu genau einem weiteren Knoten w inzident. Die Kante $\{v, w\}$ muss also entweder von v oder w überdeckt werden. Die Wahl von w für das Vertex Cover ist in jedem Fall günstiger, da dadurch noch weitere (zu w inzidente) Kanten überdeckt werden können. Löschen wir schließlich alle Kanten, die zu w inzident sind[1], so erhalten wir einen Baum kleinerer Größe, und können das gleiche Verfahren iterativ fortsetzen.

Es folgt, da wir in jedem Schritt immer den günstigeren Knoten wählen, dass das Ergebnis ein Vertex Cover minimaler Größe ist. Die Blattknoten können mit einer Tiefensuche in $O(m + n) = O(m)$ Zeit gefunden werden, der Algorithmus hat also lineare Laufzeit in Abhängigkeit der Kanten. Siehe auch Algorithmus 3 für eine Beschreibung in Pseudo–Code.

[1] Die Knoten v und w können, müssen aber wegen unserer Definition von *Blatt* nicht, gelöscht werden.

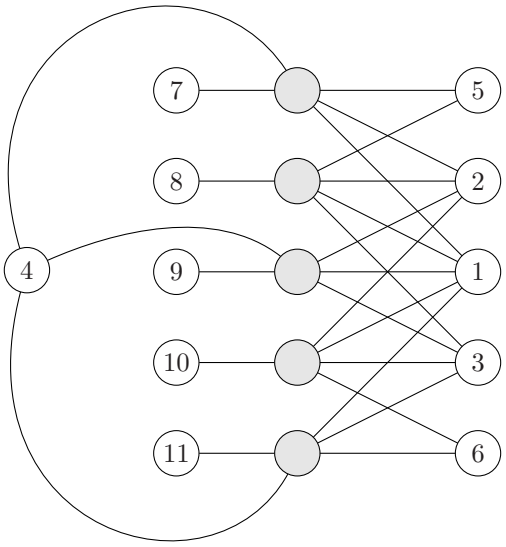

Abbildung 2: Gegenbeispiel für Aufgabe 3 (b).

Algorithmus 3 : Optimales Vertex Cover für Bäume

Eingabe : Baum $G = (V, E)$
Ausgabe : Optimales Vertex Cover

1 $C \leftarrow \emptyset$
2 **solange** $|E| \geq 1$ **tue**
3 $v \leftarrow$ Blattknoten in G
4 $w \leftarrow$ Zu v adjazenter Knoten in G
5 $C \leftarrow C \cup \{w\}$
6 Lösche w und alle zu w inzidenten Kanten
7 **return** C

Problem 4: euklid-TSP-Approximation 1pt

Algorithmus 4 : E-TSP-Approx-1(G)

1 $T \leftarrow p_1$
2 **solange** $|T| < |V|$ **tue**
3 $v, p_{i_j} \leftarrow$ Knoten mit $d(p_{i_j}, v)$ minimal, mit $p_{i_j} \in T, v \in V \setminus T$
4 $k \leftarrow |T|$
5 $T \leftarrow (p_{i_1}, \ldots, p_{i_j}, v, p_{i_{j+1}}, \ldots p_{i_k})$ //füge v zwischen p_{i_j} und $p_{i_{j+1}}$ ein
6 $(p_{i_1}, \ldots, p_{i_{k+1}}) \leftarrow T$
7 return T

Zeile 3 von E-TSP-Approx ist eine Anwendung der grünen Regel auf den Schnitt $(T, V \setminus T)$, vgl Prim. Sei $c(E) := \sum_{(i,j)\in E} d(p_i.p_j)$ Kantengewichtsfunktion, M der MST, wie er von Prim gefunden wird, sei T^* die optimale Rundtour und W eine Pre+Postorder-Traversierung von M, d.h. eine Rundtour, die jede Kante des MST zweimal betritt.
Da T^* durch Löschen einer beliebigen Kante zu einem Spannbaum wird, gilt: $c(M) \leq c(T^*)$. Da W jede Kante von M genau zweimal benutzt, gilt: $c(W) = 2c(M)$. Da die Dreiecksungleichung gilt, kann W einfach verkürzt werden, indem bereits besuchte Knoten aus der Rundtour gelöscht werden. Genau dies tut (implizit) Zeile 5, indem es eine Preorder-Traversierung aufbaut. Daher gilt: $c(T) \leq c(W)$.
Insgesamt also: $c(T) \leq c(W) = 2c(M) \leq 2c(T^*)$. Somit ist E-TSP-Approx ein 2-Approximationsalgorithmus.