Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
Dr. Thomas Worsch, Dr. Simon Gog
Demian Hespe, Yaroslav Akhremstev

2. Übungsblatt zu Algorithmen II im WS 2017/2018

http://algo2.iti.kit.edu/AlgorithmenII_WS17.php
{hespe,sanders,simon.gog,worsch,yaroslav.akhremtsev}@kit.edu

Musterlösungen

Aufgabe 1 (*Ungerade Knoten im MST*)

Für die 3/2-Approximation des metric TSP-Problems aus der Übung wurde ein minimales perfektes Matching auf den Knoten mit ungeradem Grad in einem MST verwendet. Das ist nur möglich, wenn eine gerade Anzahl an Knoten in dieser Menge sind. Beweisen Sie, dass dies der Fall ist.

Musterlösung:
Es ist zu zeigen, dass die Anzahl an Knoten mit ungeradem Grad gerade ist. In der Tat ist dies für jeden Graphen der Fall. Formaler:

$$\sum_{v \in V} deg(v) = 2|E|,$$

also eine gerade Zahl.
Wir betrachten nun die Parität dieser Summe, also ob sie gerade oder ungerade ist. Die Addition gerade Zahlen ändert nichts an der Parität der Summe (eine ungerade Zahl plus eine gerade ist immer noch ungerade - ebenso für gerade Zahlen). Damit die Summe von n ungeraden Zahlen gerade wird, muss n gerade sein (die Summe einer geraden und einer ungeraden Zahl ist eine ungerade Zahl - die Summe zweier ungeraden Zahlen ist gerade). Damit muss die Anzahl der ungeraden Summenglieder oben gerade sein.

Aufgabe 2 (*TSP zu metric TSP*)

Geben Sie eine Polynomialzeitreduktion vom allgemeinen TSP-Problem auf das metric TSP-Problem an. Die Menge der optimalen TSP-Touren in beiden Graphen soll dabei die gleiche sein. Warum widerspricht Ihre Reduktion nicht dem Beweis, dass es keine constant factor Approximation in Polynomialzeit für das allgemeine TSP-Problem gibt? (Falls $P \neq NP$)

Musterlösung:
Gegeben sei ein Graph $G = (V, E)$ mit Kantengewichtsfunktion w, die die Dreiecksungleichung nicht erfüllt. Gesucht ist eine neue Gewichtsfunktion w', die die Dreiecksungleichung erfüllt, aber alle optimalen TSP-Touren für w beibehält und keine neuen optimalen Touren hinzufügt.

Wir definieren $w'(e) = w(e) + M$ mit $M = \max_{e \in E} w(e)$. Damit erhöht sich der Wert *jeder* TSP-Tour um nM, womit die Menge der optimalen TSP-Touren gleich bleibt.
Außerdem gilt:

$$w'(a,b) + w'(b,c) = w(a,b) + w(b,c) + 2M \geq 2M \geq w(a,c) + M = w'(a,c),$$

womit w' die Dreiecksungleichung erfüllt ist.

Aus der Vorlesung wissen wir, dass es keine constant factor approximation in Polynomialzeit für das allgemeine TSP-Problem gibt. Es existieren aber solche Approximationen für metric TSP. Gegeben eine a-Approximation für metric TSP betrachten wir, warum dies unter unserer Reduktion keine constant factor approximation für das allgemeine TSP-Problem ergibt.
Sei OPT eine optimale Lösung unter w und OPT' eine optimale Lösung unter w', also

$$w(OPT') = w(OPT) + (n-1) \cdot M$$

Eine a-Approximation für w' hätte also maximal den Wert

$$a \cdot w(OPT') = a \cdot (w(OPT) + (n-1) \cdot M) = a \cdot w(OPT) + a(n-1) \cdot M$$

Da diese Tour $n-1$ Kanten verwendet und jedes Kantengewicht um M erhöht wurde, hat diese Tour unter w das Gewicht

$$a \cdot w(OPT) + (a-1)(n-1) \cdot M$$

Da $a > 1$ ist dies keine constant factor approximation.

Aufgabe 3 (*Kleinaufgaben: Laufzeiten*)

Sei $T(n, \varepsilon)$ die Laufzeit eines Approximationsalgorithmus und $g(n, \varepsilon)$ seine Approximationsgarantie. Geben Sie für die folgenden Fälle an, ob der Algorithmus ein PTAS, FPTAS oder keines von beiden ist. Begründen Sie Ihre Antwort jeweils kurz.

- $T_1(n, \varepsilon) = \frac{1}{\varepsilon} \cdot (4n^3 + n^2)$, $g_1(n, \varepsilon) = (1 - \varepsilon)$
- $T_2(n, \varepsilon) = \frac{1}{\varepsilon} \cdot n^2$, $g_2(n, \varepsilon) = (1 + 2\varepsilon)$
- $T_3(n, \varepsilon) = \sqrt{n} + n^{\frac{3}{2}}$, $g_3(n, \varepsilon) = 2 + \frac{1}{n}$
- $T_4(n, \varepsilon) = n \cdot \log \frac{1}{\varepsilon}$, $g_4(n, \varepsilon) = (1 - \varepsilon)$
- $T_5(n, \varepsilon) = \varepsilon + e^{\log n} + n^5$, $g_5(n, \varepsilon) = (1 + \varepsilon)$
- $T_6(n, \varepsilon) = n^{\frac{1}{\varepsilon}} + n^5$, $g_6(n, \varepsilon) = (2 + \varepsilon)$

Anmerkung: Für $g_6(n, \epsilon)$ können Sie annehmen, dass der Approximationsalgorithmus mit beliebigem $\epsilon \in (-1, 0)$ spezifiziert werden kann.

Musterlösung:

a) Ein Approximationsalgorithmus wird als PTAS bezeichnet, wenn seine Laufzeit $T(n, \varepsilon)$ polynomiell in n ist und sich seine Approximationsgarantie beliebig nahe der 1 nähern kann. Für ein FPTAS muss zusätzlich $T(n, \varepsilon)$ polynomiell in $\frac{1}{\varepsilon}$ sein. Mit dieser Definition ergibt sich für die angegebenen Algorithmen:

- $T_1(n, \varepsilon) = \frac{1}{\varepsilon} \cdot (4n^3 + n^2)$, $\quad g_1(n, \varepsilon) = (1 - \varepsilon)$
  Bei dem angegebenen Algorithmus handelt es sich um ein FPTAS. $T_1(n, \varepsilon)$ hängt polynomiell von n und $\frac{1}{\varepsilon}$ ab und die Approximationsgarantie kann beliebig nahe an die 1 herankommen.
- $T_2(n, \varepsilon) = \frac{1}{\varepsilon} \cdot n^2$, $\quad g_2(n, \varepsilon) = (1 + 2\varepsilon)$
  Bei dem angegebenen Algorithmus handelt es sich um ein FPTAS. Es gilt die gleiche Begründung wie bei $T_1(n, \varepsilon)$ und $g_1(n, \varepsilon)$. Will man die klassische Approximationsgarantie ohne den Faktor 2 sehen, substituiert man einfach $\delta = 2\varepsilon$.
- $T_3(n, \varepsilon) = \sqrt{n} + n^{\frac{3}{2}}$, $\quad g_3(n, \varepsilon) = 2 + \frac{1}{n}$
  Bei dem angegebenen Algorithmus handelt es sich weder um ein FPTAS noch um ein PTAS. Die Approximationsgarantie hängt von n ab und kann nicht beliebig nahe an 1 herankommen.
- $T_4(n, \varepsilon) = n \cdot \log \frac{1}{\varepsilon}$, $\quad g_4(n, \varepsilon) = (1 - \varepsilon)$
  Bei dem angegebenen Algorithmus handelt es sich um ein FPTAS. Die Approximationsgarantie kann sich beliebig der 1 nähern. $T_4(n, \varepsilon)$ hängt polynomiell von n und auch von $\frac{1}{\varepsilon}$ (da $\log x = O(n) = O(poly(n))$).
- $T_5(n, \varepsilon) = \varepsilon + e^{\log n} + n^5$, $\quad g_5(n, \varepsilon) = (1 + \varepsilon)$
  Bei dem angegebenen Algorithmus handelt es sich um ein FPTAS. $T_5(n, \varepsilon)$ ist polynomiell in n ($e^{\log n} = n$) und $\frac{1}{\varepsilon}$ ($\varepsilon = \frac{1}{\varepsilon}^{-1}$). Außerdem kann sich die Approximationsgarantie beliebig der 1 nähern.
- $T_6(n, \varepsilon) = n^{\frac{1}{\varepsilon}} + n^5$, $\quad g_6(n, \varepsilon) = (2 + \varepsilon)$
  Bei dem angegebenen Algorithmus handelt es sich um ein PTAS. Die Approximationsgarantie kann sich beliebig der 1 nähern und $T_6(n, \varepsilon)$ hängt zwar polynomiell von n ab, aber nicht von $\frac{1}{\varepsilon}$. Für die klassische Darstellung der Approximationsgarantie ersetzt man $\varepsilon = \delta - 1$.

Aufgabe 4 (*Analyse+Rechnen: Vertex-Cover*)

Gegeben sei ein ungerichteter Graph $G = (V, E)$. Sei $N(v) = \{w|(v, w) \in E\}$ die Menge aller Nachbarn von Knoten v. Man definiert folgende Teilmengen der Knotenmenge V des Graphen:
Vertex Cover (VC).

- Ein VC ist eine Teilmenge $C \in V$, so dass f.a. Kanten $(u, v) \in E$ mindestens einer ihrer Endpunkte in C enthalten ist. Üblicherweise ist ein minimales VC gesucht.

Gegeben sei folgender Algorithmus zur Berechnung eines *vertex cover* C für einen Graph $G = (V, E)$:

1. Initialisiere die Ergebnismenge $C = \varnothing$ als leere Menge.
2. Wähle Kante $(u, v) \in E$ beliebig.
3. Füge u, v zu C hinzu.
4. Entferne u, v und alle inzidenten Kanten aus G.
5. Wiederhole solange G noch Kanten hat

Nach Abschluss des Algorithmus ist $C \subseteq V$ ein *vertex cover*, d.h. für jede Kante $(u, v) \in E$ ist einer ihrer beiden Knoten in C. Falls o.b.d.A. $u \in C$ sagt man auch Knoten u *überdeckt* Kante (u, v).

a) Zeigen Sie, dass der angegebene Algorithmus ein korrektes *vertex cover* berechnet.

b) Geben Sie ein Beispiel an, in dem der Algorithmus ein minimales *vertex cover* liefert.

c) Geben Sie ein Beispiel an, in dem der Algorithmus kein minimales *vertex cover* liefert.

d) Zeigen oder widerlegen Sie, dass der Algorithmus eine 2-Approximation für *vertex cover* berechnet, d.h. dass er höchstens doppelt soviele Knoten auswählt als minimal nötig.

Betrachten Sie abschließend diesen alternativen Algorithmus zur Bestimmung einer 2-Approximation von *vertex cover*:

1. Initialisiere die Ergebnismenge $C = \varnothing$ als leere Menge.
2. Wähle Knoten $u \in V$ mit minimalem Grad.
3. Füge u zu C hinzu.
4. Entferne u und alle inzidenten Kanten aus G
5. Wiederhole solange G noch Kanten hat

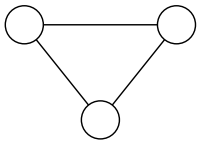
Der Algorithmus berechnet offenbar –mit ähnlichen Argumenten wie in Teilaufgabe (a)– ein *vertex cover*. Es bleibt folgende Frage zu beantworten:

e) Zeigen oder widerlegen Sie, dass der Algorithmus eine 2-Approximation für *vertex cover* berechnet, d.h. dass er höchstens doppelt soviele Knoten auswählt als minimal nötig.

Musterlösung:

a) Nach Abschluss enthält der Graph keine Kanten mehr. Da eine Kante nur entfernt wird, wenn einer ihrer Knoten in C aufgenommen wurde, ist folglich jede Kante $e \in E$ von einem Knoten überdeckt. Damit ist C ein *vertex cover*.

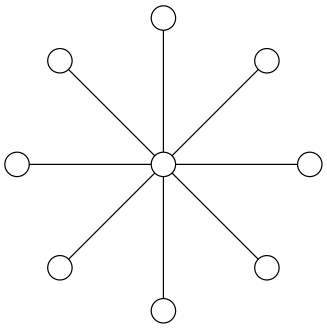
b) In folgendem Graphen werden immer genau zwei Knoten ausgewählt. Dies ist optimal, da einzelner Knoten nur zwei der drei Kanten überdecken.



c) In folgendem Graphen werden immer zwei Knoten ausgewählt. Das ist nicht optimal, da ein einzelner Knoten genügen würde.

d) Sei A die Menge der in Schritt 2 ausgewählten Kanten. Es gilt $|C| = 2|A|$, da beide Knoten jeder ausgewählten Kante zu C hinzugefügt und anschließend zusammen mit allen inzidenten Kanten aus G entfernt werden, so dass sie nicht noch einmal ausgewählt werden können. Damit folgt auch, dass keine zwei Kanten aus A einen Knoten gemeinsam haben können. Sei nun C^* ein minimales *vertex cover*. C^* enthält nach Definition mindestes einen Knoten jeder Kante, also insbesondere einen Knoten jeder Kante aus A. Da keine zwei Kanten aus A vom gleichen Knoten aus C^* überdeckt werden können, gilt $|C^*| \geq |A|$. Es folgt $|C| = 2|A| \leq 2|C^*|$. Die Lösung C des angegebenen Algorithmus ist also maximal doppelt so groß wie die optimale Lösung C^*. Damit berechnet der Algorithmus eine 2-Approximation.

e) Der Algorithmus berechnet keine 2-Approximation. Folgender Graph ist ein Gegenbeispiel:



Ein optimales *vertex cover* benötigt nur den Knoten in der Mitte. Der angegebene Algorithmus markiert allerdings $n - 1$ Knoten (entweder alle Randknoten, oder den mittleren Knoten und alle Randknoten bis auf einen).

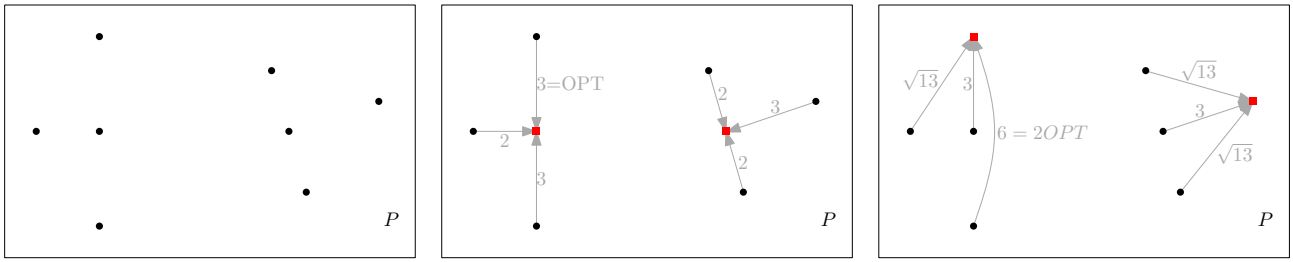
Aufgabe 5 (*Analyse: Metrisches k-Zentren Problem (*)*)

Gegeben sei eine Menge an Punkten $P \subset \mathbb{R}^2$ in der Ebene sowie eine Zahl $k > 0$. Gesucht ist eine k-elementige Teilmenge $K \subset P$ dieser Punkte, genannt Zentren, so dass für jeden Punkt $p \in P$ der maximale Abstand zu seinem nächstgelegenen Zentrum minimal ist.

Es existiert folgender *greedy* Algorithmus, der eine 2-Approximation des Problems berechnet:

1. Wähle beliebigen Punkt aus P als erstes Zentrum
2. Wähle Punkt aus P als nächstes Zentrum mit größter Entfernung zu allen bisherigen Zentren (d.h. der den maximalen kürzesten Abstand zu einem Zentrum besitzt)
3. Wiederhole bis k Zentren gewählt worden sind

Das folgende Beispiel veranschaulicht die Problemstellung für $k = 2$:



Links ist eine Punktmenge P abgebildet. In der Mitte ist eine optimale Lösung zu sehen. Die roten Quadrate sind die ausgewählten Zentren. Die Kanten geben das nächstgelegene Zentrum für jeden Knoten sowie den Abstand an. Rechts ist eine weitere aber suboptimale Lösung aufgezeigt.

Zunächst einige allgemeine Fragen zu diesem Algorithmus:

a) Beschreiben Sie in Worten, welche Bedeutung OPT sowie die Aussage eine Lösung sei eine 2-Approximation des metrischen k-Zentren Problems, haben.

b) Handelt es sich bei dem angegebenen Algorithmus um ein PTAS, ein FPTAS oder um keines von beiden. Begründen Sie kurz.

Im Folgenden soll gezeigt werdem, dass der Algorithmus tatsächlich eine 2-Approximation berechnet. Dafür sind zunächst einige Vorüberlegungen nötig.

c) Zeigen Sie, bei einer Auswahl von $k + 1$ Punkten aus P existieren immer mindestens 2 Punkte, die das gleiche nächstgelegene Zentrum haben.

d) Gegeben eine optimale Lösung, wie groß kann der Abstand zwischen zwei Punkten maximal sein, wenn diese das gleiche nächstgelegene Zentrum besitzen?

e) In einer Lösung des *greedy* Algorithmus sei der maximale Abstand eines Punktes $p \notin K$ zu seinem nächstgelegenen Zentrum $> l$. Zeigen Sie, dass l eine untere Schranke für den Abstand zwischen je zwei der Zentren $k_i,\ k_j \in K,\ i \neq j$ der Lösung darstellt.

f) Zeigen Sie mit obigen Aussagen, dass der angegebene *greedy* Algorithmus eine 2-Approximation für das Problem berechnet. Nehmen Sie dazu an, in der Lösung des Algorithmus sei der maximale Abstand eines Punktes $p \notin K$ zu seinem nächstgelegenen Zentrum $> 2 \cdot OPT$, und führen Sie diese Aussage zum Widerspruch.

Hinweis: Machen Sie zunächst eine Aussage über die paarweisen Abstände von $k + 1$ speziell gewählten Punkten. Verwenden Sie anschließend einen Vergleich zu Abständen in der optimalen Lösung, um zum Widerspruch zu gelangen.

Musterlösung:

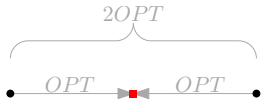
a) Im metrischen k-Zentren Problem charakterisiert OPT den maximalen Abstand eines Punktes zu seinem nächstgelegenen Zentrum. Eine 2-Approximation bedeutet, dass der Abstand eines Punktes zu seinem nächstgelegenen Zentrum höchstens doppelt so groß ist wie OPT (das rechte Bild in der Aufgabenstellung beschreibt eine 2-Approximation).

b) Der beschriebene *greedy* Algorithmus ist weder ein PTAS noch ein FPTAS, da sich seine Approximationsgüte nicht beliebig der 1 annähern lässt (bei polynomieller Laufzeit ist der Algorithmus immerhin in APX).

c) Angenommen k Punkte haben paarweise verschiedene nächstgelegene Zentren. Damit ist jeder Punkt einem anderen Zentrum zugeordnet und alle Zentren sind verwendet. Ein weiterer Punkt hätte auf alle Fälle ein schon verwendetes Zentrum als nächstgelegenes Zentrum. Wäre dies nicht der Fall, so gäbe es mehr als k Zentren oder die bisherigen Punkte hätten nicht alle paarweise verschiedene nächstgelegene Zentren.

   Dieses Prinzip wird auch *pigeon hole principle* genannt.

d) Wie in der Abbildung zu sehen, können sich beide Punkte auf entgegengesetzten Seiten des Zentrums befinden mit maximalem Abstand OPT. Damit ist ihr Abstand zueinander $2 \cdot OPT$.



   (Hinweis: Für diese Aussage wurde der metrische Raum benötigt!)

e) Nach Voraussetzung ist Abstand $d(p, k) > l$ f.a. $k \in K$ und damit auch f.a. $k \in K/\{k_j\}$. Angenommen es gelte für einen Abstand $d(k_i, k_j) \leq l$. O.b.d.A. werde k_i vor k_j als Zentrum ausgwählt. Dann würde im weitern Verlauf des Algorithmus p anstatt k_j als Zentrum gewählt werden, da p den größeren minimalen Abstand zu den bisherigen Zentren hat. Da aber k_j ein Zentrum ist, muss $d(k_i, k_j) > l$ gelten.

f) Angenommen, in der Lösung des Algorithmus sei der maximale Abstand eines Punktes $p \notin K$ zu seinem nächstgelegenen Zentrum $> 2 \cdot OPT$. Nach Teilaufgabe (e) wäre der Abstand zwischen allen Zentren $> 2 \cdot OPT$. Das würde bedeuten, es gäbe $k + 1$ Punkte mit einem paarweisen Abstand $> 2 \cdot OPT$ (Punkt p sowie die k Zentren). Wähle zwei dieser Punkte, die in einer optimalen Lösung das gleiche nächste Zentrum haben. Teilaufgabe (c) belegt die Existenz dieser Punkte. Nach Teilaufgabe (d) hätten sie allerdings einen Abstand $\leq 2 \cdot OPT$. Widerspruch zu der Aussage, dass alle diese Punkte einen paarweisen Abstand $> 2 \cdot OPT$ besitzen.