L. 67

Optimierung für Studierende der Informatik

Thomas Andreae

Wintersemester 2017/18
Blatt 10

B: Hausaufgaben zum 15./16. Januar 2018

1. Wir betrachten das Lastwagenproblem aus Präsenzaufgabe 1 und verwenden die dort eingeführten Bezeichnungen.

a) Zeigen Sie die Richtigkeit der Behauptung aus Präsenzaufgabe 1b) für den Fall, dass m gerade ist.

b) Es sei $k \geq 2$ eine ganze Zahl. Zeigen Sie, dass es für jedes derartige k ein Beispiel gibt, für das $m^* = k$ und $m = 2k - 1$ gilt.
Hinweis: Lösen Sie zunächst die Fälle $k = 2, 3, 4$ und orientieren Sie sich an diesen Fällen.

c) Begründen Sie kurz, weshalb aus b) folgt, dass unser Algorithmus *kein* γ-Approximationsalgorithmus für $\gamma < 2$ ist.

---

a) Wir setzen voraus, dass <u>m gerade</u> ist. Aus der Vorgehensweise des Greedy-Algorithmus ergibt sich:

$$L_1 + L_2 > K,\ L_3 + L_4 > K, \ldots, L_{m-1} + L_m > K.$$

Durch Aufsummieren folgt

$$\sum_{i=1}^{m} L_i > \frac{m}{2} K.$$

Die Gesamtlast ist also größer als $\frac{m}{2}K$, d.h., dass $\frac{m}{2}$ Lastwagen niemals ausreichen – auch dann nicht, wenn man eine Methode verwendet, die besser als unser Greedy-Algorithmus ist.

Also: $m^* > \frac{m}{2}$ und somit $m < 2m^*$.

b) Es gebe $2k-1$ Container. Die Gewichte seien wie folgt: $w_1 = 1$, $w_2 = k$, $w_3 = 1$, $w_4 = k, \ldots, w_{2k-2} = k$, $w_{2k-1} = 1$. Die Schranke K sei gleich k.

Dann gilt $m = 2k-1$ und $m^* = k$.

c) Für die Beispielserie aus b) gilt

$$\frac{m}{m^*} = \frac{2k-1}{k} \longrightarrow 2 \text{ für } k \to \infty.$$

Zu jedem $\gamma < 2$ gibt es demnach ein k, so dass $\gamma < \frac{2k-1}{k} = \frac{m}{m^*}$ gilt. Es folgt $m > \gamma m^*$, weshalb unser Algorithmus kein γ-Approximationsalgorithmus ist.

2. a) Auf Seite 166 des Skripts wurde im Zusammenhang mit dem Knotenüberdeckungsproblem die folgende Frage gestellt: *Ist es möglich, den gefundenen Faktor 2 zu verbessern, indem man den Algorithmus unverändert lässt, aber eine raffiniertere Analyse des Algorithmus vornimmt?* Zeigen Sie mithilfe des vollständig bipartiten Graphen $K_{n,n}$, dass dies nicht möglich ist.

a) $G=(V,E)$ sei der vollständige bipartite Graph $K_{n,n}$. Die Knotenmenge V sei die disjunkte Vereinigung von X und Y mit $|X|=|Y|=n$. Es gelte $E=\{\{x,y\}: x\in X, y\in Y\}$. Es folgt:

(1) $c(G)=n$.

<u>Beweis.</u> X (oder auch Y) ist eine Knotenüberdeckung mit n Knoten. Eine Menge $A\subseteq V$ mit $|A|<n$ kann niemals eine Knotenüberdeckung von G sein, da es wegen $|A|<n$ ein $x_0\in X$ und ein $y_0\in Y$ gibt mit $x_0\notin A$ und $y_0\notin A$. Es folgt, dass die Kante $\{x_0,y_0\}$ nicht von A getroffen wird. Also gilt (1).

(2) Jedes nicht erweiterbare Matching von G besteht aus n Kanten.

<u>Beweis:</u> Ist M ein Matching von G mit $|M|<n$, so gibt es ein $x_0\in X$ und ein $y_0\in Y$, so dass weder x_0 noch y_0 auf einer Kante von M liegt. Also lässt sich M durch Hinzunahme von $\{x_0,y_0\}$ erweitern. Folglich gilt (2).

Aus (2) folgt: Das vom Algorithmus gelieferte U besteht aus sämtlichen Knoten von G, d.h. $|U| = 2n$. Also gilt wegen (1):

Lässt man den Algorithmus unverändert, so lässt sich der Gütegarantiefaktor 2 nicht verbessern.

b) Es sei $k \geq 2$ eine fest gewählte ganze Zahl. Gegeben sei eine Menge S und eine Kollektion $T_1, \ldots, T_m$ von k-elementigen Teilmengen von S. Es gelte also

$$T_i \subseteq S \text{ und } |T_i| = k \quad (i = 1, \ldots, m).$$

Eine Teilmenge $H \subseteq S$ wird ein *Hitting Set* genannt, falls alle Mengen T_i von H getroffen werden, d.h., falls $H \cap T_i \neq \emptyset$ für alle $i = 1, \ldots, m$ gilt. Gesucht ist ein Hitting Set mit einer minimalen Anzahl von Elementen. (Mitteilung: Dies ist ein NP-schweres Optimierungsproblem.) Wir nennen das beschriebene Problem k-HITTING SET. Das Problem lässt sich also wie folgt beschreiben:

k-HITTING SET

Eingabe: eine Menge S sowie eine Kollektion von k-elementigen Teilmengen von S.

Gesucht: ein Hitting Set mit einer minimalen Anzahl von Elementen.

Man beachte: k ist *nicht* Teil der Eingabe, sondern eine *Konstante*.

(i) Beschreiben Sie einen k-Approximationsalgorithmus für k-HITTING SET.

(ii) Weisen Sie nach, dass der von Ihnen unter (i) vorgeschlagene Algorithmus tatsächlich ein k-Approximationsalgorithmus ist.

Hinweis zu (i) und (ii): Denken Sie zunächst an den Spezialfall $k = 2$. Sie kennen bereits einen 2-Approximationsalgorithmus für diesen Spezialfall: siehe Abschnitt 11.7.1 im Skript.

Im Folgenden wird

(i) ein Algorithmus für k-HITTING SET beschrieben,

(ii) der Nachweis erbracht, dass es sich bei dem beschriebenen Algorithmus um einen k-Approximationsalgorithmus handelt.

(i) Wir konstruieren eine Menge $\mathcal{M} \subseteq \{T_1, \dots, T_m\}$ wie folgt: Man startet mit $\mathcal{M} = \emptyset$. Solange es ein $T_j \notin \mathcal{M}$ gibt, das zu allen Elementen von $\mathcal{M}$ disjunkt ist, wähle man ein solches T_j beliebig aus und füge es zu $\mathcal{M}$ hinzu.

Auf diese Art erhält man am Ende eine Menge $\mathcal{M}$ von disjunkten T_i, die nicht erweiterbar ist. Das bedeutet: Zu jedem $T_j \notin \mathcal{M}$ gibt es ein $T_i \in \mathcal{M}$, so dass $T_j \cap T_i \neq \emptyset$ gilt. Mit $\mathcal{M}$ bezeichnen wir ab jetzt eine wie beschrieben konstruierte nicht erweiterbare Teilmenge von $\{T_1, \dots, T_m\}$.

Es sei H die Menge der Elemente von S, die in den T_i aus $\mathcal{M}$ vorkommen. Wir wählen H als Output unseres Approximationsalgorithmus, d.h., H soll das gewünschte Hitting Set sein.

Der Nachweis, dass H tatsächlich ein Hitting Set ist und dass es sich beim beschriebenen Algorithmus um einen k-Approximationsalgorithmus handelt, wird in (ii) erbracht.

(ii) Es gelte $|\mathcal{M}| = t$. Dann folgt $|H| = kt$.
Da $\mathcal{M}$ nicht erweiterbar ist, wird jedes T_j von H getroffen, d.h., H ist ein Hitting Set.

Es sei nun H^* ein Hitting Set mit einer minimalen Anzahl von Elementen. Wir haben zu zeigen:

$$|H| \leq k|H^*|. \qquad (*)$$

Dies folgt so: Da die Mengen aus $\mathcal{M}$ disjunkt sind, gilt

$$|\mathcal{M}| \leq |H^*|.$$

Es folgt wegen $|\mathcal{M}| = t$ und $|H| = kt$:

$$|H| = kt = k|\mathcal{M}| \leq k|H^*|.$$

Dies zeigt (*), womit nachgewiesen ist, dass es sich bei unserem Algorithmus um einen k-Approximationsalgorithmus handelt. $\square$