Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Übungsblatt 5

Vorlesung Theoretische Grundlagen der Informatik im WS 19/20

Ausgabe: 17. Dezember 2019
Abgabe: 14. Januar 2020, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

Aufgabe 1

(1 + 2 = 3 Punkte)

(a) Sei Π ein $\mathcal{NP}$-vollständiges Problem, zu dem ein pseudopolynomialer Algorithmus existiert. Warum impliziert dies nicht die Existenz eines pseudopolynomialen Algorithmus für jedes $\mathcal{NP}$-vollständige Problem?

(b) Zeigen Sie, dass ein stark $\mathcal{NP}$-vollständiges Problem genau dann von einem pseudopolynomialen Algorithmus entschieden wird, wenn $\mathcal{P} = \mathcal{NP}$ gilt.

Aufgabe 2

(2 + 2 = 4 Punkte)

Gegeben sei ein ungewichteter, ungerichteter Graph $G = (V, E)$. Für zwei Knoten $u, v \in V$ bezeichnen wir mit $\texttt{dist}(u, v)$ die Länge des kürzesten Pfads zwischen u und v in G. Der *Durchmesser* von G ist definiert als $D(G) = \max_{u,v \in V} \texttt{dist}(u, v)$. Also ist der Durchmesser die Länge des längsten kürzesten Pfads in G.

(a) Beschreiben Sie, wie der Durchmesser eines Graphen in Polynomialzeit berechnet werden kann.

(b) Zeigen Sie, dass in Zeit $\mathcal{O}(|V|+|E|)$ eine 2-Approximation für den Durchmesser eines Graphen berechnet werden kann.

Aufgabe 3

(1 + 1 + 2 + 2 + 2 + 1 = 9 Punkte)

In der Übung wurde das $\mathcal{NP}$-vollständige Entscheidungsproblem SetCover (deutsch *Mengenüberdeckung*) vorgestellt. Hier betrachten wir das entsprechende Optimierungsproblem min-SetCover:

Gegeben: Universum $\mathcal{U} = \{u_1, \ldots, u_m\}$, Menge $\mathcal{S} = \{S_1, \ldots, S_n\}$ von Teilmengen $S_i \subseteq \mathcal{U}$
Gesucht: Menge $C \subseteq \{1, \ldots, n\}$ mit $|C|$ minimal, sodass $\bigcup_{i \in C} S_i = \mathcal{U}$ gilt.

In der Vorlesung haben wir gesehen, dass sich jedes $\mathcal{NP}$-vollständige Problem als *ganzzahliges Programm* (engl. *Integer Program*, IP) darstellen lässt. Im Folgenden ist ein noch unvollständiges IP für min-SetCover gegeben, das wir mit SC-$\mathbb{N}$ bezeichnen:

Gegeben:
MIN-SETCOVER-Instanz $I = (\mathcal{U}, \mathcal{S})$
Variablen:
Für jede Teilmenge $S_i \in \mathcal{S}$ eine Variable x_i
Nebenbedingungen:

(I) Für jede Teilmenge $S_i \in \mathcal{S}$: $x_i \in \{0, 1\}$

(II) Für jedes Element $u_i \in \mathcal{U}$: $\square$

Zielfunktion:
Minimiere $f_{\mathbb{N}}(I) = \sum_{i=1}^{n} x_i$

Aus einer Lösung für SC-$\mathbb{N}$ lässt sich eine Lösung für MIN-SETCOVER konstruieren, indem man $C = \{i \in \{1, \ldots, n\} \mid x_i = 1\}$ setzt. Der Wert von x_i gibt also an, ob S_i in der Mengenüberdeckung enthalten ist oder nicht.

(a) Ergänzen Sie den fehlenden Teil von Nebenbedingung (II), sodass eine Lösung von SC-$\mathbb{N}$ einer Lösung von MIN-SETCOVER entspricht.

Eine gängige Methode, IPs zu approximieren, ist die *LP-Relaxierung*. Dabei wird die Beschränkung aufgehoben, dass die Variablen ganzzahlige Werte haben müssen. Das dadurch entstehende *lineare Programm* (LP) lässt sich in Polynomialzeit lösen. Im Fall von SETCOVER erhalten wir das Problem SC-$\mathbb{R}$. Der einzige Unterschied gegenüber SC-$\mathbb{N}$ ist in Nebenbedingung (I): Die x_i dürfen nun beliebige reelle Zahlen aus dem Intervall $[0, 1]$ sein. Dementsprechend ist auch der Wert der Zielfunktion $f_{\mathbb{R}}(I) = \sum_{i=1}^{n} x_i$ reellwertig.

Sei im Folgenden $\mathrm{OPT}_{\mathbb{R}}(I)$ der Wert von $f_{\mathbb{R}}(I)$ für eine optimale Lösung von SC-$\mathbb{R}$ und $\mathrm{OPT}_{\mathbb{N}}(I)$ der Wert von $f_{\mathbb{N}}(I)$ einer optimalen Lösung von SC-$\mathbb{N}$.

(b) Zeigen Sie, dass für jede MIN-SETCOVER-Instanz $I = (\mathcal{U}, \mathcal{S})$ gilt: $\mathrm{OPT}_{\mathbb{R}}(I) \leq \mathrm{OPT}_{\mathbb{N}}(I)$.

(c) Geben Sie eine MIN-SETCOVER-Instanz $I = (\mathcal{U}, \mathcal{S})$ an, für die $\mathrm{OPT}_{\mathbb{R}}(I) < \mathrm{OPT}_{\mathbb{N}}(I)$ gilt. Geben Sie für beide Probleme eine optimale Lösung an.

Wir betrachten nun folgenden Algorithmus $\mathcal{A}$, der eine (nicht notwendigerweise optimale) Lösung für SC-$\mathbb{N}$ berechnet:

1. Berechne eine optimale Lösung $X = (x_1, \ldots, x_n)$ mit $x_i \in [0, 1]$ für SC-$\mathbb{R}$.
2. Sei f die maximale Anzahl von Mengen S_i, in denen irgendein Element u_i vorkommt.
3. Generiere eine Lösung $X' = (x'_1, \ldots, x'_n)$ mit $x'_i \in \{0, 1\}$ für SC-$\mathbb{N}$, indem wir jedes x'_i wie folgt wählen:

$$x'_i = \begin{cases} 1 & \text{falls } x_i \geq \frac{1}{f} \\ 0 & \text{sonst.} \end{cases}$$

(d) Zeigen Sie, dass Algorithmus $\mathcal{A}$ eine Lösung für SC-$\mathbb{N}$ berechnet, also dass die Nebenbedingungen erfüllt sind.

(e) Zeigen Sie, dass für beliebige MIN-SETCOVER-Instanzen I gilt: $\mathcal{A}(I) \leq f \cdot \mathrm{OPT}_{\mathbb{N}}(I)$

(f) Ist $\mathcal{A}$ ein Approximationsalgorithmus für SC-$\mathbb{N}$ mit relativer Gütegarantie? Begründen Sie!

Aufgabe 4

(2 + 3 = 5 Punkte)

Das Optimierungsproblem BinPacking ist wie folgt definiert:

Gegeben: Endliche Menge $M = \{a_1, \ldots, a_n\}$ mit Gewichtungsfunktion $s : M \to (0, 1]$.
Gesucht: Zuweisung der Elemente in M zu einer minimalen Anzahl an Bins m, sodass für jeden Bin B_i mit $i = 1, \ldots, m$ gilt:

$$\sum_{a \in B_i} s(a) \leq 1$$

In der Übung wurde der Algorithmus NextFit vorgestellt. Dabei werden die Elemente in der Reihenfolge $a_1, a_2, \ldots, a_n$ bearbeitet. Passt das Element a_i noch in den aktuellen Bin, so wird es in diesen eingefügt. Ansonsten wird ein neuer Bin hinzugefügt und a_i in diesen eingefügt. In der Übung wurde gezeigt, dass NextFit ein Approximationsalgorithmus ist und eine obere Schranke von 2 für dessen relative Güte bewiesen.

(a) Zeigen Sie, dass diese obere Schranke gewissermaßen optimal ist. Geben Sie dazu für jedes $k > 0$ eine Folge von Elementen an, sodass NextFit bei Abarbeitung dieser Folge $2k$ Bins benötigt, ein optimaler Algorithmus dagegen nur $k + 1$. Daraus folgt, dass die relative Gütegarantie mindestens $\lim_{k\to\infty} \frac{2k}{k+1} = 2$ sein muss.

(b) Die Strategie, um Elemente in die Bins einzufügen, wird nun verändert. Statt nur den aktuellen Bin zu betrachten, wird jetzt ein Element in den ersten Bin eingefügt, in dem noch ausreichend Platz ist. Diese Strategie wird FirstFit genannt. Zeigen Sie, dass für diesen neuen Approximationsalgorithmus $\mathcal{A}$ gilt: $\mathcal{R}_{\mathcal{A}} \geq \frac{5}{3}$.

Hinweis: Finden Sie drei Mengen von jeweils 6 *Elementen gleicher Größe, sodass der Algorithmus die ersten* 6 *Elemente in einen Bin, die nächsten* 6 *Elemente in* 3 *Bins und die letzten* 6 *Elemente in* 6 *Bins einfügt. In einer optimalen Lösung sollten alle Elemente in* 6 *Bins eingefügt werden können.*

Aufgabe 5

(2 + 2 + 2 = 6 Punkte)

Betrachten Sie folgendes Job-Scheduling-Problem: Gegeben sind n Jobs $J = \{j_1, \ldots, j_n\}$, die auf einer Maschine bearbeitet werden sollen. Die Maschine kann immer nur einen Job gleichzeitig bearbeiten. Jeder Job j_i hat eine *Bearbeitungszeit* $p_i > 0$ und einen *Freigabezeitpunkt* $r_j \geq 0$, vor dem er nicht bearbeitet werden darf. Gesucht ist ein *Scheduling* S, das für jeden Zeitpunkt angibt, welcher Job gerade bearbeitet wird. Es soll

$$\sum_{i=1}^{n} C_i^S$$

minimiert werden, wobei wir mit C_i^S den Zeitpunkt bezeichnen, zu dem Job j_i im Scheduling S fertiggestellt wird.

Wir unterscheiden zwei Arten von Scheduling: Beim *nicht-präemptiven* Scheduling muss ein einmal angefangener Job solange bearbeitet werden, bis er fertiggestellt ist. Beim *präemptiven* Scheduling darf ein bereits angefangener Job unterbrochen und durch einen anderen ersetzt werden.

Betrachten Sie folgenden Algorithmus für die präemptive Variante des Problems:

Algorithmus 1: PreemptiveScheduling

```
for τ ← 0, 1, ... do
    if J = ∅ then break
    J_r ← {j_i ∈ J | r_i ≤ τ}
    if J_r = ∅ then
        S(τ) = ⊥
    else
        Wähle j_i ∈ J_r mit p_i minimal
        S(τ) ← j_i
        p_i ← p_i − 1
        if p_i = 0 then J ← J \ {j_i}
```

Es wird also in jedem Zeitschritt unter allen verfügbaren Jobs derjenige mit der geringsten verbleibenden Bearbeitungszeit ausgewählt und bearbeitet. Falls gerade kein Job verfügbar ist, befindet sich die Maschine für diesen Schritt im Leerlauf.

(a) Zeigen Sie, dass PreemptiveScheduling eine optimale Lösung berechnet.

Wir betrachten nun folgenden Algorithmus $\mathcal{A}$ für die nicht-präemptive Variante des Problems: Führe zunächst PreemptiveScheduling aus. Dies generiert ein präemptives Scheduling O. Sei $j_1, j_2, \ldots, j_n$ die Reihenfolge, in der die Jobs in O fertiggestellt werden, d.h. $C_1^O < C_2^O < \cdots < C_n^O$. Bearbeite die Jobs in dieser Reihenfolge, d.h. bearbeite Job j_i, sobald er freigegeben wurde und Job j_{i-1} fertiggestellt wurde.

(b) Sei S das von $\mathcal{A}$ erstellte Scheduling. Sei τ_i^S der Zeitpunkt, zu dem S mit der Bearbeitung von Job j_i beginnt. Zeigen Sie: Zu keinem Zeitpunkt zwischen C_i^O und τ_i^S ist die Maschine im Leerlauf.

(c) Zeigen Sie, dass $\mathcal{A}$ ein Approximationsalgorithmus mit relativer Gütegarantie 2 ist. Zeigen Sie dafür, dass für jeden Job j_i gilt: $C_i^S \leq 2 \cdot C_i^O$.

Aufgabe 6

(3 + 2 = 5 Punkte)

(a) Sei $k > 1$ beliebig. Zeigen Sie: Wenn es einen Approximationsalgorithmus mit relativer Gütegarantie k^2 für max-Clique gibt, dann gibt es auch einen Approximationsalgorithmus mit relativer Gütegarantie k.

Hinweis: Betrachten Sie für einen gegebenen Graphen $G = (V, E)$ den Graphen $G^2 = (V \times V, E')$, wobei $\{(u, v), (w, x)\} \in E'$ genau gilt, wenn die folgenden beiden Bedingungen erfüllt sind:

- $\{u, w\} \in E$ *oder* $u = w$
- $\{v, x\} \in E$ *oder* $v = x$

(b) Angenommen, MAX-CLIQUE hätte einen Approximationsalgorithmus mit relativer Gütegarantie k für irgendein $k > 1$. Zeigen Sie, dass es dann ein PTAS für MAX-CLIQUE gäbe. Ist Ihr Algorithmus auch ein FPTAS?

Anmerkung: In der Tat lässt sich zeigen, dass es für MAX-CLIQUE *keinen relativen Approximationsalgorithmus gibt, und somit auch kein PTAS.*

Aufgabe 7

(2 + 1 = 3 Punkte)

Sei p ein Polynom und Π ein $\mathcal{NP}$-schweres Minimierungsproblem, bei dem die Optimierungsfunktion f_Π des Problems ganzzahlig ist. Außerdem gelte für jede Instanz I, dass $\mathrm{OPT}_\Pi(I) < p(|I_u|)$, wobei I_u die unäre Kodierung von I bezeichnet.

Zeigen Sie:

(a) Falls es für Π ein FPTAS gibt, so gibt es auch einen pseudopolynomialen Algorithmus für Π.

(b) Falls Π stark $\mathcal{NP}$-vollständig ist und $\mathcal{P} \neq \mathcal{NP}$ gilt, gibt es für Π kein FPTAS.