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

Übungsblatt 5

Vorlesung Theoretische Grundlagen der Informatik im WS 17/18

Ausgabe 20. Dezember 2017
Abgabe 16. Januar 2018, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

Aufgabe 1 (2 Punkte)

Zeigen Sie, dass das TravelingSalesman-Problem stark NP-vollständig ist.

Lösung:
Sei $G = (V, E)$ eine Instanz des NP-vollständigen HamiltonianCycle-Problems. Setze $E' = V \times V$ und definiere für $e \in E'$

$$c(e) = \begin{cases} 1 & \text{falls } e \in E \\ 2 & \text{sonst} \end{cases}$$

Dann enthält G einen Hamiltonkreis genau dann, wenn $((V, E'), c)$ eine Traveling-Salesman-Tour der Länge $k = |V|$ enthält. Es werden nur die Zahlen Eins, Zwei und k kodiert. Die Größe all dieser Zahlen ist linear in der Größe von V. Damit ist das TravelingSalesman-Problem selbst bei unärer Kodierung von $((V, E'), c, k)$ NP-vollständig.

Aufgabe 2 (1 + 1 + 2 + 1 + 1 = 6 Punkte)

Aus Vorlesung und Übung kennen Sie das NP-vollständige Problem VertexCover:

> **Gegeben:** ungerichteter Graph $G = (V, E)$ und $k \in \mathbb{N}$.
> **Gesucht:** Menge $V' \subseteq V$ mit $|V'| \leq k$, sodass $\forall (u, v) \in E : u \in V' \vee v \in V'$.

(a) Ist das Problem in der oben gegebenen Formulierung ein Optimierungsproblem, ein Optimalwertproblem, oder ein Entscheidungsproblem? Geben Sie auch die beiden anderen Formulierungen an.

Bemerkung: Die Aufgabenstellung ergibt in dieser Form keinen Sinn, siehe unten.

Es soll eine approximative Lösung für das VertexCover Problem berechnet werden. Dazu wird zunächst $V' = \emptyset$ gesetzt. Solange G eine Kante enthält wird dann eine beliebige Kante $(u, v) \in E$ gewählt, beide Knoten u, v zu V' hinzugefügt und anschließend aus G entfernt. Um einen Knoten w aus G zu entfernen werden alle zu w inzidenten Kanten und w selbst entfernt.

(b) Zeigen Sie, dass dieser Algorithmus tatsächlich ein VertexCover berechnet.

(c) Zeigen Sie, dass dieser Algorithmus ein Approximationsalgorithmus für das VertexCover Problem mit einer relativen Gütegarantie von 2 ist. *Hinweis:* Die Menge der gewählten Kanten bildet ein Matching.

(d) Zeigen Sie, dass unter der Annahme $\mathsf{P} \neq \mathsf{NP}$ kein Polynomialzeitapproximationsalgorithmus mit absoluter Gütegarantie für das VertexCover Problem existiert.

Der Algorithmus soll angepasst werden, um eine approximative Lösung für das IndependentSet Problem zu finden.

> **Gegeben:** ungerichteter Graph $G = (V, E)$.
>
> **Gesucht:** kardinalitätsmaximale Menge $V' \subseteq V$, für die aus $v \in V'$ für alle Kanten $\{u, v\} \in E$ folgt, dass $u \notin V'$.

Dazu wird anstatt der Menge V' die Menge $V \setminus V'$ ausgegeben.

(e) Ist der so abgewandelte Algorithmus ein Approximationsalgorithmus für das Independent-Set Problem mit relativer Gütegarantie 2? Begründen Sie!

Lösung:

(a) Die Formulierung in der Aufgabenstellung ist unsinnig. Deshalb lässt sich die Frage am treffendsten mit „keines davon“ beantworten.

Das Entscheidungsproblem wäre:

> **Gegeben:** ungerichteter Graph $G = (V, E)$ und $k \in \mathbb{N}$.
>
> **Gefragt:** existiert eine Menge $V' \subseteq V$ mit $|V'| \leq k$, sodass für alle Kanten $(u, v) \in E$ gilt: $u \in V' \vee v \in V'$?

Die Antwort wäre dann also „ ja“ oder „nein“. Das Optimalwertproblem ist:

> **Gegeben:** ungerichteter Graph $G = (V, E)$.
>
> **Gesucht:** wie lautet das minimale $k \in \mathbb{N}$, sodass eine Menge $V' \subseteq V$ mit $|V'| = k$ exisitiert, sodass für alle Kanten $(u, v) \in E$ gilt: $u \in V' \vee v \in V'$.

Die Antwort wäre dann also eine natürliche Zahl. Das Optimierungsproblem ist:

> **Gegeben:** ungerichteter Graph $G = (V, E)$.
>
> **Gesucht:** kardinalitätsminimale Menge $V' \subseteq V$, sodass für alle Kanten $(u, v) \in E$ gilt: $u \in V' \vee v \in V'$.

Die Anwort wäre dann also eine Menge von Knoten.

(b) Betrachte eine beliebige Kante (u, v). Da der Algorithmus erst terminiert, wenn G keine Kante mehr enthält, wird (u, v) irgendwann aus G entfernt. Dies geschieht nur dann, wenn entweder u oder v aus G entfernt wird. Dann gilt $u \in V' \vee v \in V'$.

(c) Betrachte die Menge von Kanten, die der Algorithmus auswählt. Da für jede solche Kante (u, v) danach sowohl u als auch v aus G entfernt werden, kann danach keine zu (u, v) adjazente Kante mehr gewählt werden. Die Menge der gewählten Kanten bilden also ein Matching. Jedes VertexCover muss (u, v) abdecken und deshalb u oder v enthalten. Die Menge V' enthält also höchstens doppelt so viele Elemente, wie eine optimale Lösung.

(d) Angenommen, es gäbe einen solchen Algorithmus $\mathcal{A}$ mit absoluter Gütegarantie k. Bei Eingabe einer Instanz $G = (V, E)$ wird dann einfach eine Instanz generiert, die aus $k + 1$ Kopien von G besteht. Da $\mathcal{A}$ eine absolute Gütegarantie k hat, muss mindestens eine Kopie optimal gelöst worden sein. Gibt man also die beste Teilkopie zurück, hat man in Polynomialzeit eine optimale Lösung von G berechnet. Dies ist unter Annahme von $\mathsf{P} \neq \mathsf{NP}$ ein Widerspruch zur NP-Vollständigkeit von VertexCover.

(e) Nein. Betrachte dazu eine Menge von unabhängigen Kanten. Der Algorithmus gibt dann die leere Menge aus, was keine 2-Approximation eines IndependentSet ist.

Aufgabe 3

(1 + 3 + 1 = 5 Punkte)

Gegeben sei eine Grundmenge $S = \{1, \ldots, n\}$ und ein Mengensystem $\mathcal{C} \subseteq 2^S$. Jede Menge $A \in \mathcal{C}$ enthalte maximal 3 Elemente, d.h. $|A| \leq 3$. Ein HittingSet für $\mathcal{C}$ ist eine Teilmenge $X \subseteq S$, so dass für jedes $A \in \mathcal{C}$ gilt $A \cap X \neq \emptyset$. Gesucht ist ein HittingSet minimaler Kardinalität, was ein NP-vollständiges Problem ist. Sie sollen deshalb nun versuchen, einen Approximationsalgorithmus zu entwickeln.

In einem ersten Versuch wählt man dazu immer aus einer noch nicht abgedeckten Menge ein beliebiges Element a und fügt es dem HittingSet hinzu. Betrachten Sie den folgenden Algorithmus:

Algorithmus 1: 3-HittingSet-Approximation, Versuch 1

Eingabe: $S = \{1, \ldots, n\}$, $\mathcal{C} \subseteq 2^S$ mit $|A| \leq 3$ für alle $A \in \mathcal{C}$
Ausgabe: HittingSet X für $\mathcal{C}$

$X \leftarrow \emptyset$;
solange $\mathcal{C} \neq \emptyset$ **tue**
  $A \leftarrow$ wähle eine beliebige Menge $A \in \mathcal{C}$;
  $a \leftarrow$ wähle ein beliebiges Element aus A;
  $X \leftarrow X \cup \{a\}$;
  $\mathcal{C} \leftarrow \mathcal{C} \setminus \{A \in \mathcal{C} \mid A \cap X \neq \emptyset\}$;

(a) Zeigen Sie, dass Algorithmus 1 kein Approximationsalgorithmus mit konstanter oder relativer Gütegarantie ist.

Im zweiten Versuch wird die Strategie auf eine Art und Weise angepasst, die vielleicht verschwenderisch erscheint. Anstatt nur ein Element a zu X hinzuzufügen, werden *alle* Elemente aus A zu X hinzugefügt. Betrachten Sie den folgenden Algorithmus:

Algorithmus 2: 3-HittingSet-Approximation

Eingabe: $S = \{1, \ldots, n\}$, $\mathcal{C} \subseteq 2^S$ mit $|A| \leq 3$ für alle $A \in \mathcal{C}$
Ausgabe: HittingSet X für $\mathcal{C}$

$X \leftarrow \emptyset$;
solange $\mathcal{C} \neq \emptyset$ **tue**
  $A \leftarrow$ wähle eine Menge $A \in \mathcal{C}$;
  $X \leftarrow X \cup A$;
  $\mathcal{C} \leftarrow \mathcal{C} \setminus \{A \in \mathcal{C} \mid A \cap X \neq \emptyset\}$;

(b) Zeigen Sie, dass Algorithmus 2 ein 3-Approximationsalgorithmus ist, d.h. dass

$$\mathcal{A}(I) \leq 3 \cdot \mathrm{OPT}(I)$$

gilt. Hierbei sei I eine beliebige Instanz des Problems, $\mathcal{A}(I)$ die Kardinalität des von Algorithmus 2 zurückgegebenen HittingSets, sowie $\mathrm{OPT}(I)$ die Kardinalität einer optimalen Lösung.

(c) Zeigen Sie, dass die relative Gütegarantie von Algorithmus 2 nicht besser als 3 ist. Geben Sie dazu eine Familie von Mengensystemen $\mathcal{C}$ an, für die Algorithmus 2 ein HittingSet berechnet, dass dreimal so viele Elemente enthält wie eine optimale Lösung.

Lösung:

(a) Sei $S = \{1, 2, \ldots, n\}$ und $\mathcal{C} = \{\{1, 2\}, \{1, 3\}, \ldots, \{1, n\}\}$. Dann wählt Algorithmus 1 ggf. immer $a \neq 1$, was $|X| = n - 1$ impliziert. Eine optimale Lösung würde hingegen nur aus dem Element 1 bestehen.

(b) In der ersten Zeile wird immer eine Menge A gewählt, so dass gilt $A \cap X = \emptyset$, sonst wäre A bereits aus $\mathcal{C}$ entfernt worden. Sei X^* eine beliebige aber feste optimale Lösung. Weil X^* ein HittingSet ist, gilt $A \cap X^* \neq \emptyset$. Kombiniert ergibt sich $A \cap (X^* \setminus X) \neq \emptyset$.

Pro Schleifendurchlauf wird also mindestens ein Element aus X^* zu X hinzugefügt, welches davor noch nicht zu X gehörte. Spätestens nach $|X^*|$ Schleifendurchläufen wird X also ein HittingSet sein. Da für alle $A \in \mathcal{C}$ gilt $|A| \leq 3$ werden pro Schleifendurchlauf außerdem höchstens drei Elemente zu X hinzugefügt. Insgesamt gilt dann $|X| \leq 3|X^*|$. Damit ist $\mathcal{A}$ ein 3-Approximationsalgorithmus für das HittingSet-Problem.

(c) Sei $\mathcal{C} = \{\{1, 2, 3\}, \{4, 5, 6\}, \{7, 8, 9\}, \ldots\}$. Eine optimale Lösung würde $n/3$ Elemente enthalten, Algorithmus 2 berechnet aber eine Lösung mit n Elementen.

Aufgabe 4

(1 + 2 + 2 + 3 + 2 = 10 Punkte)

Betrachten Sie das folgende Problem. Gegeben sind m identische Maschinen $M = \{i_1, \ldots, i_m\}$ und eine Menge von n Jobs $J = \{j_1, \ldots, j_n\}$. Jede Maschine kann immer nur einen Job bearbeiten und jeder Job muss ohne Unterbrechung auf einer Maschine bearbeitet werden. Die Bearbeitungszeit von Job j ist gegeben durch $p_j \geq 0$ und ist unabhängig von der Maschine auf der j bearbeitet wird.

Gesucht ist eine Zuweisung der Jobs auf die Maschinen, so dass die Gesamtbearbeitungszeit minimiert wird. Ist beispielsweise jeder Maschine $i \in M$ die Jobmenge $J_i \subseteq J$ zugeordnet, so muss gelten, dass $J_1 \cup J_2 \cup \cdots \cup J_m = J$ und $J_i \cap J_{i'} = \emptyset$ für $i \neq i'$. Die Gesamtbearbeitungszeit ist dann gegeben durch

$$\max_{i \in M} T_i \quad \text{wobei} \quad T_i = \sum_{j \in J_i} p_j.$$

Betrachten Sie nun den folgenden Algorithmus $\mathcal{A}$ um eine Zuweisung der Jobs auf Maschinen zu berechnen:

- Gehe durch die Jobs $j_1, \ldots, j_n$ in dieser Reihenfolge und weise den nächsten Job der Maschine zu, die momentan die geringste Bearbeitungszeit hat.

Betrachten Sie nun den Algorithmus $\tilde{\mathcal{A}}$ der wie $\mathcal{A}$ funktioniert, außer dass die Jobs anfangs nach nicht-aufsteigender Bearbeitungszeit sortiert werden.

- Sortiere die Jobs nach nicht-aufsteigender Bearbeitungszeit, d.h. $p_{j_1} \geq p_{j_2} \geq \cdots \geq p_{j_n}$.
- Gehe durch die Jobs $j_1, \ldots, j_n$ in dieser Reihenfolge und weise den nächsten Job der Maschine zu, die momentan die geringste Bearbeitungszeit hat.

Bezeichne T die Gesamtbearbeitungszeit in von $\mathcal{A}$ gefundenen Zuweisung, $\tilde{T}$ die Gesamtbearbeitungszeit der von $\tilde{\mathcal{A}}$ gefundenen Zuweisung und T^* die Gesamtbearbeitungszeit in einer optimalen Zuweisung.

(a) Beweisen Sie, dass $T^* \geq \max_{j \in J} p_j$ und $T^* \geq \frac{1}{m} \sum_{j \in J} p_j$.

(b) Beweisen Sie, dass $\mathcal{A}$ ein Approximationsalgorithmus mit relativer Gütegarantie von höchstens 2 ist (d.h. $\mathcal{R}_{\mathcal{A}}^{\infty} \leq 2$).
Hinweis: Vergleichen Sie T mit den unteren Schranken an T^* aus a.

(c) Finden Sie für jedes $\varepsilon > 0$ Instanzen für die gilt $T \geq (2 - \varepsilon)T^*$.

(d) Beweisen Sie, dass $\tilde{\mathcal{A}}$ ein Approximationsalgorithmus mit relativer Gütegarantie von höchstens 3/2 ist (d.h. $\mathcal{R}_{\tilde{\mathcal{A}}}^{\infty} \leq 3/2$).
Hinweis: Finden Sie eine weitere untere Schranke an T^* für den Fall dass $n > m$.

(e) Finden Sie für jedes $\varepsilon > 0$ Instanzen für die gilt $\tilde{T} \geq (4/3 - \varepsilon)T^*$.

Bemerkung: Man kann zeigen, dass $\mathcal{R}_{\tilde{\mathcal{A}}}^{\infty} \leq 4/3$, also $\tilde{\mathcal{A}}$ ein 4/3-Approximationsalgorithmus ist, aber das müssen Sie hier nicht zeigen.

Lösung:

(a) Natürlich muss der Job j' mit der längsten Bearbeitungszeit, d.h. $p_{j'} = \max_{j \in J} p_j$, auch bearbeitet werden. Wenn $j' \in J_i$ (j' wird von Maschine i bearbeitet), dann gilt $T^* \geq T_i = \sum_{j \in J_i} p_j \geq p_{j'} = \max_{j \in J} p_j$.
Weiterhin folgt aus $T_i \leq T^*$, dass $\sum_{i \in M} T_i \leq mT^*$ und mit $J_1 \cup \cdots \cup J_m = J$ folgt $mT^* \geq \sum_{i \in M} T_i = \sum_{i \in M} \sum_{j \in J_i} p_j = \sum_{j \in J} p_j$.

(b) Wir zeigen, dass $T \leq \max_{j \in J} p_j + \frac{1}{m} \sum_{j \in J} p_j$. Mit den Ungleichungen in (a) folgt dann $T \leq 2T^*$ für alle Instanzen und demnach $\mathcal{R}_{\mathcal{A}}^{\infty} \leq 2$.

Betrachte also eine Maschine i mit $T_i = T$, also eine mit längster Bearbeitungszeit in der von $\mathcal{A}$ berechneten Zuweisung. Sei j' der letzte Job der Maschine i zugewiesen wurde. Das heißt zu diesem Zeitpunkt hatte Maschine i die momentan geringste Bearbeitungszeit, also höchstens die momentan durchschnittliche Bearbeitungszeit. Es gilt also

$$T_i = (T_i - p_{j'}) + p_{j'} \leq \frac{1}{m} \sum_{j \in J} p_j + \max_{j \in J} p_j.$$

(c) Wir betrachten m Maschinen, $m(m-1)$ Jobs j mit Bearbeitungszeit $p_j = 1$ und einen Job j' mit Bearbeitungszeit $p_{j'} = m$. Nun lasse Algorithmus $\mathcal{A}$ zuerst alle Jobs mit Bearbeitungszeit 1 zuweisen. Es ist klar, dass dann jeder Maschine genau $m - 1$ Jobs zugewiesen werden. Am

Ende wird Job j' dann einer Maschine i zugewiesen und wir haben $T = T_i = m - 1 + m = 2m - 1$.

Andererseits wird in einer optimalen Zuweisung Job j' von einer Maschine i' bearbeitet und die restlichen $m - 1$ Maschinen bearbeiten je m der übrigen $m(m - 1)$ Jobs. Das gibt eine Gesamtbearbeitungszeit von $T^* = m$.

Also gilt $T = 2T^* - 1 = (2 - 1/m)T^*$. Wir wählen also $m \geq 1/\varepsilon$.

(d) Falls $n \leq m$, so besteht eine optimale Zuweisung aus einem Job pro Maschine und diese Zuweisung wird auch von $\tilde{\mathcal{A}}$ berechnet. In diesem Fall gilt also sogar $\tilde{T} = T^*$.

Falls nun $n > m$, so muss in einer optimalen Zuweisung mindestens eine Maschine zwei Jobs aus $\{j_1, \ldots, j_{m+1}\} \subseteq J$ bearbeiten. Da j_{m+1} eine geringste Bearbeitungszeit in dieser Menge hat, gilt also $T^* \geq 2p_{j_{m+1}}$.

Betrachte nun eine Maschine i mit $T_i = \tilde{T}$ und den letzten Job j' der i von Algorithmus $\tilde{\mathcal{A}}$ zugewiesen wurde. Da die Jobs sortiert durchlaufen wurden, gilt $p_{j'} \leq p_{j_{m+1}}$ und wir erhalten analog zu (b)

$$T_i = (T_i - p_{j'}) + p_{j'} \leq \frac{1}{m} \sum_{j \in J} p_j + p_{j_{m+1}} \leq T^* + T^*/2 = \frac{3}{2} T^*.$$

(e) Wir betrachten m Maschinen und $2m+1$ Jobs mit $m \geq 3$. Drei Jobs haben eine Bearbeitungszeit von m und jeweils zwei der verbleibenden $2m - 2$ Jobs haben eine Bearbeitungszeit von $m + 1$, $m + 2$ bis $2m - 1$. Algorithmus $\tilde{\mathcal{A}}$ weist nun $m - 1$ Maschinen jeweils zwei Jobs zu mit Gesamtbearbeitungszeit $3m - 1$ und einer Maschine i drei Jobs mit Gesamtbearbeitungszeit $4m - 1$. Es gilt also $\tilde{T} = 4m - 1$.

Andererseits werden in einer optimalen Zuweisung die drei Jobs mit Bearbeitungszeit m auf einer Maschine ausgeführt und die restlichen $2m - 2$ Jobs jeweils in geeigneten Paaren auf einer der restlichen $m - 1$ Maschinen mit einer Gesamtbearbeitungszeit von $T^* = 3m$.

Es gilt also $T = 4/3T^* - 1 = (4/3 - 1/(3m))T^*$. Wir wählen also $m \geq 1/(3\varepsilon)$.

Aufgabe 5 (3 Punkte)

Zeigen Sie, dass es keinen polynomiellen Approximationsalgorithmus mit absoluter Gütegarantie für Max2Sat gibt, falls $\mathsf{P} \neq \mathsf{NP}$ gilt.

Lösung:
Angenommen, es gäbe einen polynomiellen Approximationsalgorithmus mit absoluter Gütegarantie k für Max2Sat. Daraus konstruieren wir einen polynomiellen Lösungsalgorithmus für Sat, was der Annahme $\mathsf{P} \neq \mathsf{NP}$ widerspricht.

Sei $I = (V, \mathcal{C})$ eine Instanz von Sat. Konstruiere eine Instanz $I' = (V', \mathcal{C}')$ von Max2Sat durch $k + 1$-maliges „Kopieren“ von I. Setze dazu $V' = V \times \{0, 1, \ldots, k\}$. Für jedes

$$C = (v_1 \cup v_2 \cup \ldots \cup v_s \cup \neg v_{s+1} \cup \ldots \cup \neg v_t) \in \mathcal{C}$$

fügen wir die Klauseln

$$\bigcup_{i=0}^{k} ((v_1, i) \cup (v_2, i) \cup \ldots \cup (v_s, i) \cup \neg(v_{s+1}, i) \cup \ldots \cup \neg(v_t, i))$$

zu $\mathcal{C}'$ hinzu.

Da k eine Konstante ist, ist die Größe von I' linear in der Größe von I.

Ist I lösbar ist auch I' lösbar durch $k+1$-maliges „Kopieren“ der Lösung von I. Umgekehrt sind in einer Lösung von I' höchstens k Klauseln nicht erfüllt. Da die Instanz I aber $k+1$-Mal „kopiert“ wurde, müssen in mindestens einer Kopie von I alle Klauseln erfüllt sein. Damit ist natürlich auch I lösbar.

Insgesamt ist dann SAT in Polynomialzeit lösbar, was ein Widerspruch zur Annahme $\mathsf{P} \neq \mathsf{NP}$ ist. Also kann es keinen polynomiellen Approximationsalgorithmus mit absoluter Gütegarantie für MAX2SAT geben.

Aufgabe 6

(3 + 1 = 4 Punkte)

Sei p ein Polynom und Π ein NP-schweres Minimierungsproblem, sodass 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 FPAS gibt, so gibt es auch einen pseudopolynomialen Algorithmus für Π.

(b) Unter der Annahme $\mathsf{P} \neq \mathsf{NP}$ gibt es für ein stark NP-vollständiges Problem kein FPAS.

Lösung:

(a) Sei $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$ ein FPAS für Π, d.h., die Laufzeit von $\mathcal{A}_\varepsilon$ ist polynomial in $|I|$ und $\frac{1}{\varepsilon}$. Formal: Die Laufzeit von $\mathcal{A}_\varepsilon$ ist $q(|I|, \frac{1}{\varepsilon})$, wobei q ein Polynom ist.
Setze $\varepsilon = \frac{1}{p(|I_u|)}$. Für die Lösung $\mathcal{A}_\varepsilon$ gilt.

$$\mathcal{R}_{\mathcal{A}_\varepsilon} = \frac{\mathcal{A}_\varepsilon(I)}{\mathrm{OPT}_\Pi(I)} \leq (1+\varepsilon)$$

Wir erhalten also folgende Abschätzung:

$$\mathcal{A}_\varepsilon(I) \leq (1+\varepsilon)\,\mathrm{OPT}_\Pi(I) = \mathrm{OPT}_\Pi(I) + \varepsilon\,\mathrm{OPT}_\Pi(I) < \mathrm{OPT}_\pi(I) + \varepsilon p(|I_u|) = \mathrm{OPT}_\Pi(I) + 1$$

Da Π ein Minimierungsproblem ist und die Optimierungsfunktion von Π ganzzahlig ist, gilt somit $\mathcal{A}_\varepsilon(I) = \mathrm{OPT}_\Pi(I)$. $\mathcal{A}_\varepsilon(I)$ mit $\varepsilon = \frac{1}{p(|I_u|)}$ ist also exakter Algorithmus für Π.

(b) Eine analoge Aussage kann für Maximierungsprobleme gezeigt werden. Es gilt also, dass wenn ein Problem Π ein FPAS besitzt, dann auch einen pseudopolynomialen Algorithmus. Π kann somit nicht stark NP-vollständig sein.

Aufgabe 7

(4 Punkte)

Wie jeden Dezember muss der Weihnachtsmann seinen Rentierschlitten mit Geschenken beladen. Jedes Geschenk ist quaderförmig, die Ladefläche des Schlittens ist rechtechtig mit festen Seitenlängen. Damit nichts zu Bruch kommt dürfen Geschenke prinzipiell nicht gestapelt werden. Außerdem

dürfen die Geschenke nur achsenparallel auf dem Schlitten platziert werden. Da beim Rentierschlittentransport nur der Himmel die Grenze ist, dürfen die Geschenke beliebig hoch sein. Das Ziel ist es, alle Geschenke auf dem Schlitten unterzubringen.

Weil der Weihnachtsmann in den vergangenen Jahren viele Stunden damit verbracht hat, die Geschenke möglichst geschickt auf seinem Schlitten unterzubringen, würde er gerne ein effizientes (also Polynomialzeit-) Verfahren zum Geschenkeverladen entwickeln.

Zeigen Sie, dass das unter er Annahme $\mathsf{P} \neq \mathsf{NP}$ unmöglich ist.

Lösung:
Wir zeigen, dass GeschenkeVerladen NP-vollständig ist.

Sei die Menge $M = \{a_1, a_2, \ldots, a_n\}$ mit Gewichtungsfunktion $s : M \to \{0, 1\}$. Beim Entscheidungsproblem BinPacking geht es darum, zu entscheiden, ob sich die Elemente aus M zu k Bins der Größe Eins einteilen lassen.

Der Schlitten habe die Maße $\ell \times b$. Nehme an, dass gilt $b/k > \ell$ (ist dies nicht der Fall, kann ein geeignetes sehr großes Geschenk konstruiert werden). Konstruiere für jedes $a \in M$ ein Geschenk g mit Länge $s(a) \cdot \ell$, Breite b/k und Höhe $b+1$. Die Kantenlängenzuweisung für g legt die Orientierung des Geschenks auf dem Schlitten vollständig fest. Dadurch induziert jede Platzierung der Geschenke eine Lösung der BinPacking-Instanz und umgekehrt.