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

Übungsblatt 5

Vorlesung Theoretische Grundlagen der Informatik im WS 20/21

Ausgabe: 12. Januar 2021
Abgabe: 26. Januar 2021, 11:30 Uhr (digital im ILIAS)

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?

Lösung:
Bei der polynomialen Transformation einer Eingabeinstanz I in eine Instanz I' beschränken wir die Länge $|I'|$ polynomial in der Länge von I. Dies impliziert allerdings nicht, dass auch die größte Zahl $\max(I')$ aus I' polynomial durch I' beschränkt ist. Dies führt unmittelbar dazu, dass die Unärkodierung der Instanz I' nicht polynomial beschränkt ist in der Unärkodierung von I.

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

Lösung:
Angenommen, es existiert ein pseudopolynomialer Algorithmus $\mathcal{A}$ für ein stark $\mathcal{NP}$-vollständiges Problem. Dann ist die Anzahl der Bits in der Unärkodierung durch ein Polynom beschränkt, da $\max(I)$ polynomial beschränkt ist. Mit Hilfe von $\mathcal{A}$ kann somit jedes Problem in $\mathcal{NP}$ in Polynomialzeit entschieden werden.

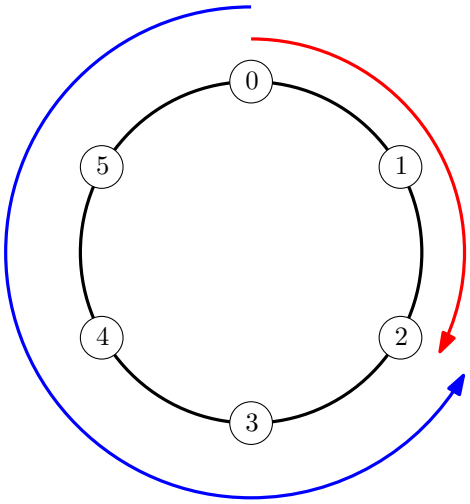
Angenommen, es gilt $\mathcal{P} = \mathcal{NP}$. Die stark $\mathcal{NP}$-vollständigen Probleme sind eine Teilmenge der $\mathcal{NP}$-vollständigen Probleme. Es existiert also ein Algorithmus $\mathcal{A}$, der polynomiale Laufzeit in n benötigt. Es bleibt zu zeigen, dass $\mathcal{A}$ auch pseudopolynomial ist. Sei die Zahl n die Anzahl der Bits in der Binärkodierung (oder ein Kodierungsschema, das polynomial transformierbar in die Binärkodierung ist). Die Anzahl der Bits N einer Unärkodierung ist somit $N \in \Theta(2^n)$. Angenommen, der Algorithmus ist nicht polynomial in N, dann folgt sofort ein Widerspruch zur polynomialen Laufzeit in n. Jeder Polynomialzeitalgorithmus ist also auch ein pseudopolynomialer Algorithmus.

Aufgabe 2

(2 + 1 + 1 + 1 + 1 + 2 = 8 Punkte)

Das Optimierungsproblem RingRouting tritt in Telekommunikationsnetzwerken auf. Gegeben sind n Knoten, die ringförmig durch ungerichtete Kanten verbunden sind. Formal lässt sich der Ring als Graph mit Knotenmenge $V = \{0, 1, \ldots, n-1\}$ und Kantenmenge $E = \{\{i, (i+1) \mod n\} \mid$

$i \in V\}$ darstellen. Außerdem gegeben ist eine Menge C von m *Nachrichten* der Form (i, j), die von Startknoten i zu Zielknoten j geroutet werden sollen. Es gibt zwei Möglichkeiten, eine Nachricht zu routen: mit dem Uhrzeigersinn oder gegen den Uhrzeigersinn über den Ring. Folgendes Beispiel zeigt einen Ring mit 6 Knoten und die beiden Möglichkeiten, die Nachricht $(0, 2)$ zu routen:



Die *Last* $L(e)$ einer Kante e ist die Anzahl an Nachrichten, die über e geroutet werden. Die *Gesamtlast* $L = \max_{e \in E} L(e)$ ist die höchste Last unter allen Kanten im Ring. Ziel des Optimierungsproblems RingRouting ist es, die Nachrichten so zu routen, dass die Gesamtlast minimiert wird. Es kann gezeigt werden, dass die Entscheidungsvariante von RingRouting $\mathcal{NP}$-vollständig ist.

In der Vorlesung haben wir gesehen, dass sich jedes $\mathcal{NP}$-vollständige Problem als *ganzzahliges lineares Programm* (engl. *Integer Linear Program*, ILP) darstellen lässt. Um RingRouting als ILP darzustellen, führen wir für jede Nachricht c eine Variable $x_c \in \{0, 1\}$ ein. Dabei soll $x_c = 0$ bedeuten, dass die Nachricht gegen den Uhrzeigersinn geroutet wird, und $x_c = 1$, dass sie mit dem Uhrzeigersinn geroutet wird. Für eine Kante e bezeichnen wir mit $\overrightarrow{C}(e)$ die Menge der Nachrichten aus C, die über Kante e laufen, wenn man sie im Uhrzeigersinn routet. Analog ist $\overleftarrow{C}(e)$ die Menge der Nachrichten, die über e laufen, wenn man sie gegen den Uhrzeigersinn routet. Beachten Sie, dass $C = \overrightarrow{C}(e) \dot{\cup} \overleftarrow{C}(e)$ gilt.

Im Folgenden ist ein noch unvollständiges ILP für RingRouting gegeben, das wir mit RR-$\mathbb{N}$ bezeichnen:

Gegeben:
RingRouting-Instanz $I = (n, C)$

Variablen:
Für jede Nachricht $c \in C$ eine Variable x_c
Eine Variable L für die Gesamtlast

Nebenbedingungen:

(I) Für jede Nachricht $c \in C$: $x_c \in \{0, 1\}$

(II) $L \in \mathbb{N}_0$

(III) Für jede Kante $e \in E$: $L(e) \leq L$

Zielfunktion:
Minimiere L

(a) Vervollständigen Sie die ILP-Formulierung, indem sie eine Formel für die Last $L(e)$ in Abhängigkeit der Variablen x_c angeben.

(b) Zeigen Sie, dass für eine optimale Lösung von RR-$\mathbb{N}$ tatsächlich $L = \max_{e \in E} L(e)$ gilt.

Eine gängige Methode, ILPs 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 RingRouting erhalten wir das Problem RR-$\mathbb{R}$. Der einzige Unterschied gegenüber RR-$\mathbb{N}$ sind die Nebenbedingungen (I) und (II). Die Variablen x_c dürfen nun beliebige reelle Zahlen aus dem Intervall $[0, 1]$ sein. Intuitiv bedeutet das, dass die Nachrichten aufgeteilt werden dürfen – ein Teil der Nachricht wird im Uhrzeigersinn geroutet und der Rest gegen den Uhrzeigersinn. Dementsprechend sind auch die Last $L(e)$ der Kanten und die Gesamtlast L nun reellwertig.

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

(c) Zeigen Sie, dass für jede RingRouting-Instanz $I = (n, C)$ gilt: $\mathrm{OPT}_{\mathbb{R}}(I) \leq \mathrm{OPT}_{\mathbb{N}}(I)$.

(d) Geben Sie eine RingRouting-Instanz $I = (n, C)$ an, für die $\mathrm{OPT}_{\mathbb{R}}(I) < \mathrm{OPT}_{\mathbb{N}}(I)$ gilt. Geben Sie jeweils eine optimale Lösung an.

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

1. Berechne eine optimale Lösung $X = ((x_1, \ldots, x_m), L)$ mit $x_c \in [0, 1]$ und $L \in \mathbb{R}_0^+$ für RR-$\mathbb{R}$.
2. Generiere eine Lösung $X' = ((x'_1, \ldots, x'_m), L')$ mit $x'_c \in \{0, 1\}$ und $L' \in \mathbb{N}_0$ für RR-$\mathbb{N}$:
   (a) Wähle jedes x'_c wie folgt:
   $$x'_c = \begin{cases} 1 & \text{falls } x_c \geq \frac{1}{2} \\ 0 & \text{sonst.} \end{cases}$$
   (b) Wähle $L' =$ [ ]

(e) Wie muss L' gewählt werden, um eine gültige Lösung für RR-$\mathbb{N}$ um erzeugen?

(f) Zeigen Sie, dass $\mathcal{A}$ eine Approximationsalgorithmus für RR-$\mathbb{N}$ mit einer relativen Gütegarantie von 2 ist.

Lösung:

(a) $L(e) = \sum_{c \in \overrightarrow{C}(e)} x_c + \sum_{c \in \overleftarrow{C}(e)} (1 - x_c)$.

(b) Die Nebenbedingungen (III) erzwingen $\max_{e \in E} L(e) \leq L$. Da L ansonsten nicht beschränkt ist und minimiert werden soll, ist also $L = \max_{e \in E} L(e)$ eine optimale Lösung.

(c) Jede Lösung von RR-$\mathbb{N}$ erfüllt die Nebenbedingungen von RR-$\mathbb{R}$ und ist somit auch eine Lösung von RR-$\mathbb{R}$. Es kann also nicht $\mathrm{OPT}_{\mathbb{R}}(I) > \mathrm{OPT}_{\mathbb{N}}(I)$ gelten, weil $\mathrm{OPT}_{\mathbb{R}}(I)$ dann nicht optimal wäre.

(d) Für die Instanz $I = (2, \{(0,1)\})$ gibt es genau zwei ganzzahlige Lösungen: Die einzige Nachricht $(0,1)$ kann entweder im Uhrzeigersinn oder gegen den Uhrzeigersinn geroutet werden. In beiden Fällen entsteht eine Gesamtlast von 1. Im reellwertigen Fall ist $x_{(0,1)} = \frac{1}{2}$ eine optimale Lösung. Hier ergibt sich eine Gesamtlast von $\frac{1}{2}$.

(e) $L' = \max_{e \in E} \sum_{c \in \overrightarrow{C}(e)} x'_c + \sum_{c \in \overleftarrow{C}(e)} (1 - x'_c)$

(f) Es gilt für jedes x'_c:

$$x'_c = \left\{ \begin{array}{ll} 1 & \text{falls } x_c \geq \frac{1}{2} \\ 0 & \text{sonst} \end{array} \right\} \leq \left\{ \begin{array}{ll} 2x_c & \text{falls } x_c \geq \frac{1}{2} \\ 0 & \text{sonst} \end{array} \right\} \leq 2x_c$$

$$1 - x'_c = \left\{ \begin{array}{ll} 1 & \text{falls } 1 - x_c \geq \frac{1}{2} \\ 0 & \text{sonst} \end{array} \right\} \leq \left\{ \begin{array}{ll} 2(1 - x_c) & \text{falls } 1 - x_c \geq \frac{1}{2} \\ 0 & \text{sonst} \end{array} \right\} \leq 2(1 - x_c)$$

Damit gilt:

$$\begin{aligned} \mathcal{A}(I) &= \max_{e \in E} \sum_{c \in \overrightarrow{C}(e)} x'_c + \sum_{c \in \overleftarrow{C}(e)} (1 - x'_c) \\ &\leq 2 \left(\max_{e \in E} \sum_{c \in \overrightarrow{C}(e)} x_c + \sum_{c \in \overleftarrow{C}(e)} (1 - x_c) \right) = 2 \cdot \mathrm{OPT}_{\mathbb{R}}(I) \end{aligned}$$

Wegen Aufgabenteil (c) gilt also $\mathcal{A}(I) \leq 2 \cdot \mathrm{OPT}_{\mathbb{N}}(I)$.

Aufgabe 3

(3 + 1 = 4 Punkte)

Das MIN-STEINERTREE-Problem ist wie folgt definiert: Gegeben sind ein zusammenhängender, ungerichteter, gewichteter Graph $G = (V, E)$ mit Kantengewichtsfunktion $d\colon E \to \mathbb{R}_{\geq 0}$ und eine Menge von *Terminalknoten* $R \subseteq V$. Ein *Steinerbaum* ist ein Baum $T = (V', E')$ mit $R \subseteq V' \subseteq V$ und $E' \subseteq E$, also ein Baum in G, der alle Terminale enthält. Gesucht ist ein Steinerbaum mit minimalem Gewicht, also mit

$$D(T) := \sum_{e \in E'} d(e)$$

minimal.

In dieser Aufgabe betrachten wir eine spezielle Variante, das MIN-METRIC-STEINERTREE-Problem: Wir gehen davon aus, dass G ein vollständiger Graph ist, also $E = \binom{V}{2}$. Außerdem fordern wir, dass die Kanten die Dreiecksungleichung erfüllen, d.h. für jedes Tripel aus Knoten $u, v, w \in V$ gilt $d(\{u, v\}) + d(\{v, w\}) \geq d(\{u, w\})$.

(a) Wir bezeichnen mit $G[R]$ den von R induzierten Subgraph von G, also den Subgraph von R, der genau R enthält sowie alle Kanten, die zwischen Knoten aus R verlaufen. Sei T ein Steinerbaum von G. Zeigen Sie: Es existiert ein Spannbaum T' von $G[R]$ mit $D(T') \leq 2 \cdot D(T)$.

(b) Betrachten Sie folgenden Algorithmus $\mathcal{A}$ für das MIN-METRIC-STEINERTREE-Problem: Gegeben eine MIN-METRIC-STEINERTREE-Instanz $I = (G, d, R)$, berechne einen minimalen Spannbaum von $G[R]$. Zeigen Sie, dass $\mathcal{A}$ ein Approximationsalgorithmus mit relativer Gütegarantie 2 ist.

Lösung:

(a) Wir benutzen eine ähnliche Strategie wie bei der 2-Approximation für MIN-METRIC-TSP aus der Vorlesung. Traversiere T mit einer Tiefensuche von einem beliebigen Knoten w aus. Dies liefert eine Tour P, in der jede Kante von T genau zweimal benutzt wird, also gilt $D(P) \leq 2 \cdot D(T)$. Entferne bereits besuchte Knoten und Knoten aus $V \setminus R$ aus dieser Tour, indem sie durch direkte Kanten übersprungen werden. Aufgrund der Dreiecksungleichung wird das Gesamtgewicht dabei nicht größer. Entferne nun die letzte Kante der Tour. Das Ergebnis ist ein Pfad T', der jeden Knoten in R genau einmal besucht, also ein Hamilton-Pfad von $G[R]$. Offensichtlich ist T' auch ein Spannbaum und es gilt $D(T') \leq D(P) \leq 2 \cdot D(T)$.

(b) Seien $\mathcal{A}(I)$ der berechnete minimale Spannbaum, T der minimale Steinerbaum und T' der Spannbaum von $G[R]$ mit $D(T') \leq 2 \cdot D(T)$, der nach Aufgabenteil (a) existieren muss. Dann gilt: $D(\mathcal{A}(I)) \leq D(T') \leq 2 \cdot D(T)$. Damit ist die relative Güte von 2 gezeigt. Die Polynomialität folgt aus der Polynomialität der Spannbaumberechnung.

Aufgabe 4

(5 + 2 + 1 = 8 Punkte)

Wir betrachten das $\mathcal{NP}$-schwere Maximierungsproblem LONGESTPATHWITHGIVENENDPOINTS:

Gegeben: Ungerichteter Graph $G = (V, E)$ mit zwei unterschiedlichen Knoten $v, w \in V$
Gesucht: Länge (= Anzahl der Kanten) des längsten Pfades von v nach w

Nehmen Sie an, für dieses Problem gibt es einen Approximationsalgorithmus mit relativer Gütegarantie k für ein $k > 1$.

(a) Zeigen Sie, dass es einen Approximationsalgorithmus für LONGESTPATHWITHGIVENENDPOINTS mit relativer Gütegarantie $\sqrt{k}$ gibt.

Hinweis: Erzeugen Sie einen größeren Graphen, indem Sie Kanten durch Kopien des gegebenen Graphen ersetzen.

(b) Zeigen Sie, dass es ein PTAS für LONGESTPATHWITHGIVENENDPOINTS gibt. Ist Ihr Algorithmus auch ein FPTAS? Begründen Sie.

(c) Gibt es ein FPTAS für LONGESTPATHWITHGIVENENDPOINTS?

Lösung:

(a) Sei $\mathcal{A}$ ein Approximationsalgorithmus mit relativer Gütegarantie k. Wir konstruieren daraus einen Approximationsalgorithmus $\mathcal{A}'$ mit relativer Gütegarantie $\sqrt{k}$.

Gegeben sei ein Graph $G = (V, E)$. Wir konstruieren daraus den Graphen $G' = (V', E')$, indem wir jede Kante $xy \in E$ durch eine Kopie von G ersetzen, wobei x mit dem Knoten v der Kopie identifiziert wird und y mit w. Um zu entscheiden, welcher Endpunkt der Kante xy mit v beziehungsweise w der Kopie identifiziert wird, verwenden wir eine beliebige Ordnung

$\prec$ auf V. Wir definieren G' durch

$$\begin{aligned} V' &= V \cup \{(u,e) \mid e \in E, u \in V \setminus \{v,w\}\} \text{ und} \\ E' &= \{\{(u,e),(u',e)\} \mid u,u' \in V \setminus \{v,w\}, uu' \in E \text{ und } e \in E\} \\ &\quad \cup \{\{x,(u,e)\} \mid vu \in E, e = xy \text{ für ein } y \in V \text{ und } x \prec y\} \\ &\quad \cup \{\{(u,e),y\} \mid uw \in E, e = xy \text{ für ein } x \in V \text{ und } x \prec y\}. \end{aligned}$$

Für Knoten $x, y \in V$ bezeichnen wir den Graphen, der durch x, y und alle $(u, xy) \in V'$ in G' induziert wird, als *Kopie von* G. Die Knoten x und y bezeichnen wir als die *Endpunkte der Kopie*.

Wir zeigen: G enthält genau dann einen Pfad mit Endpunkten v und w der Länge mindestens ℓ, wenn G' eine Pfad der Länge mindestens ℓ^2 zwischen v und w enthält. Insbesondere folgt daraus $\mathrm{OPT}(G)^2 = \mathrm{OPT}(G')$.

Sei $P \subseteq G$ ein Pfad der Länge mindestens ℓ. Die Knoten von P sind insbesondere Knoten von G'. Nach Konstruktion von G' gibt es zwischen je zwei aufeinander folgenden Knoten in P einen Pfad aus mindestens ℓ Kanten. Durch Konkatenation der Pfade erhalten wir einen Pfad der Länge mindestens ℓ^2 in G'.

Sei nun $P' \subseteq G'$ ein Pfad der Länge mindestens ℓ^2. Wir betrachten die Folge S der Knoten aus $P' \cap G$, in der Reihenfolge, in der sie auf P' von v nach w auftreten. Nach Konstruktion von G' induzieren die Knoten aus S einen Pfad in G. Enthält S also mindestens $\ell+1$ Knoten, haben wir einen Pfad der Länge mindestens ℓ in G. Nehmen wir also an, dass S höchstens ℓ Knoten enthält. Die Folge S induziert in G einen Pfad der Länge höchstens $\ell - 1$, d.h. in G' einen Pfad, der nur Knoten aus höchstens $\ell - 1$ Kopien verwendet. Da P' die Länge ℓ^2 hat, liegen mindestens $\ell^2/(\ell-1) \geq \ell$ Kanten von P' in der selben Kopie G^* von G. Wir erhalten mit $P' \cap G^*$ einen Pfad der Länge mindestens ℓ zwischen den Endpunkten von G^*. Da G^* und G isomorph sind, gibt es einen Pfad aus mindestens ℓ Kanten in G.

Wende nun $\mathcal{A}$ auf G' an. Dies liefert einen Pfad von v nach w in G' der Länge $\mathrm{OPT}(G')/k$. Extrahiere daraus den entsprechenden Pfad der Länge $\sqrt{\mathrm{OPT}(G')/k} = \mathrm{OPT}(G)/\sqrt{k}$ in G. Dies liefert eine $\sqrt{k}$-Approximation. Die Konstruktion von G', und damit auch die Laufzeit, sind polynomial.

(b) Durch Iterieren des Vorgehens aus Aufgabenteil (a) lässt sich aus einem Approximationsalgorithmus mit relativer Güte k ein Approximationsalgorithmus mit relativer Güte $k^{1/i}$ für jedes $i \in \mathbb{N}$ konstruieren. Wenn eine $(1+\varepsilon)$-Approximation gewünscht ist, muss man also i so wählen, dass $k^{1/i} \leq 1 + \varepsilon$ gilt. Das ist der Fall für $i = \lceil 1/(\log_k(1+\varepsilon)) \rceil$.

Dabei wird ein Graph der Größe $\Theta(n^i) = \Theta(n^{1/\log \varepsilon})$ konstruiert, also ist der Algorithmus kein FPTAS.

(c) Die Lösungen von LongestPathWithGivenEndpoints sind ganzzahlig und polynomial in der Eingabegröße, da sie durch die Anzahl der Knoten beschränkt sind. Nach Vorlesung (schauen Sie sich den Beweis dazu noch einmal an!) gibt es also kein FPTAS, es sei denn $\mathcal{P} = \mathcal{NP}$.

Falls Sie gezeigt haben, dass aus der Annahme der Aufgabe $\mathcal{P} = \mathcal{NP}$ folgt, können Sie auch mit "Ja" antworten.

Aufgabe 5 (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.

Lösung:

(a) Sei $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$ ein FPTAS 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.

Gegeben eine Instanz I berechnet der pseudopolynomiale Algorithmus $\varepsilon = \frac{1}{p(|I_u|)}$ und wendet $\mathcal{A}_\varepsilon$ auf I an.[1] Für die Lösung $\mathcal{A}_\varepsilon(I)$ 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 ein exakter Algorithmus für Π.

Eine analoge Aussage kann auch für Maximierungsprobleme gezeigt werden.

(b) Nach Aufgabenteil (a) gilt: Wenn Π ein FPTAS besitzt, dann auch einen pseudopolynomialen Algorithmus. Π kann somit nicht stark $\mathcal{NP}$-vollständig sein.

Aufgabe 6 (2 + 1 + 1 + 1 + 2 + 1 = 8 Punkte)

Bei dem Optimierungsproblem Max-Sat ist eine Variablenmenge $V = \{v_1, v_2, \ldots, v_n\}$ und eine Klauselmenge $C = \{c_1, c_2, \ldots, c_m\}$ gegeben. Gesucht ist eine Wahrheitsbelegung der Variablen aus V so, dass möglichst viele Klauseln aus C erfüllt sind.

Betrachten Sie folgenden *randomisierten* Algorithmus für Max-Sat. Wähle eine zufällige Wahrheitsbelegung, indem jede Variable $v \in V$ unabhängig jeweils mit Wahrscheinlichkeit $1/2$ auf wahr bzw. falsch gesetzt wird. Bezeichne mit W die Anzahl an erfüllten Klauseln. Beachten Sie, dass W eine Zufallsvariable ist.

(a) Zeigen Sie $\mathbb{E}(W) = \sum_{c \in C} \left(1 - \left(\frac{1}{2}\right)^{|c|}\right)$, wobei $|c|$ die Anzahl an Literalen der Klausel c bezeichnet.

(b) Zeigen Sie $\mathbb{E}(W) \geq \frac{1}{2}\,\mathrm{OPT}$.

[1] Dazu muss zunächst eine Beschreibung von $\mathcal{A}_\varepsilon$ generiert werden. Dies muss in polynomieller Zeit in $|I|$ und $\frac{1}{\varepsilon} = p(|I_u|)$ (also polynomiell in $|I_u|$) geschehen, damit der Algorithmus pseudopolynomial bleibt. In der Vorlesung wurde ein Approximationsschema als eine Familie von Algorithmen $\{\mathcal{A}_\varepsilon \mid \varepsilon > 0\}$ definiert. Diese Definition erfordert nicht, dass sich für ein bestimmtes ε eine Beschreibung des dazugehörigen Algorithmus $\mathcal{A}_\varepsilon$ in polynomieller Zeit generieren lässt. In der Fachliteratur ist jedoch folgende Definition gängiger: Ein Approximationsschema ist ein einzelner Algorithmus $\mathcal{A}(\varepsilon)$, bei dem der Approximationsgrad $\varepsilon > 0$ eine zusätzliche Eingabe ist. Bei einem FPTAS muss dementsprechend die Laufzeit dieses einzelnen Algorithmus polynomiell in $|I|$ und $\frac{1}{\varepsilon}$ sein. Hier muss also nicht mehr das passende $\mathcal{A}_\varepsilon$ ausgewählt werden, sondern ε wird einfach als Eingabe an $\mathcal{A}(\varepsilon)$ weitergereicht.

(c) Zeigen Sie $\mathbb{E}(W) = \frac{7}{8}m$, wenn jede Klausel genau drei (unterschiedliche) Literale enthält.

Sie haben gezeigt, dass der randomisierte Max-Sat Algorithmus "erwartet" ein Approximationsalgorithmus ist. Wir *derandomisieren* den Algorithmus, um einen deterministischen, also "richtigen" Approximationsalgorithmus zu erhalten. Dazu benutzen wir bedingte Wahrscheinlichkeiten. Aufgrund der Konstruktion der zufälligen Wahrheitsbelegung gilt

$$\mathbb{E}(W) = \frac{1}{2}\mathbb{E}(W \mid v_1 \leftarrow \text{wahr}) + \frac{1}{2}\mathbb{E}(W \mid v_1 \leftarrow \text{falsch})\,.$$

Wir wählen nun $\varphi(v_1) \in \{\text{wahr}, \text{falsch}\}$ so, dass $\mathbb{E}(W \mid v_1 \leftarrow \varphi(v_1)) \geq \mathbb{E}(W \mid v_1 \leftarrow \neg\varphi(v_1))$ gilt.

(d) Zeigen Sie $\mathbb{E}(W \mid v_1 \leftarrow \varphi(v_1)) \geq \mathbb{E}(W)$.

(e) Erklären Sie, wie $\mathbb{E}(W \mid v_1 \leftarrow \text{wahr})$ und $\mathbb{E}(W \mid v_1 \leftarrow \text{falsch})$ in Polynomialzeit berechnet werden können.

Hinweis: Denken Sie an Teilaufgabe (a). Was gilt für eine Klausel, die durch die Wahrheitsbelegung $v_1 \leftarrow$ wahr erfüllt wird? Was gilt für eine Klausel, in der v_1 vorkommt, die aber durch die Wahrheitsbelegung $v_1 \leftarrow$ wahr nicht erfüllt wird? Was gilt für eine Klausel, in der v_1 nicht vorkommt?

(f) Definieren Sie $\varphi(v_i)$ für $2 \leq i \leq n$ so, dass $\mathbb{E}(W \mid v_i \leftarrow \varphi(v_i) \text{ für } 1 \leq i \leq n) \geq \mathbb{E}(W)$ gilt. Vergegenwärtigen Sie sich außerdem, dass diese Wahrheitsbelegung in Polynomialzeit berechnet werden kann.

Damit folgt, dass der derandomisierte Algorithmus ein 2-Approximationsalgorithmus für Max-Sat ist. Hat jede Klausel genau drei (unterschiedliche) Literale, so ist der derandomisierte Algorithmus sogar ein 7/8-Approximationsalgorithmus. Johan Håstad hat gezeigt[2], dass für dieses Problem kein besserer Approximationsalgorithmus exisiert, es sei denn $\mathsf{P} = \mathsf{NP}$. In diesem Sinne ist der obige Approximationsalgorithmus optimal.

Lösung:

(a) Seien $\varphi_1, \varphi_2, \ldots, \varphi_k$ alle möglichen Wahrheitsbelegungen.

Definiere für $c \in C$ und $1 \leq i \leq k$:

$$\delta(c, \varphi_i) = \begin{cases} 1 & \text{falls } \varphi_i \text{ die Klausel } c \text{ erfüllt} \\ 0 & \text{sonst} \end{cases}$$

Dann

$$\begin{aligned} \mathbb{E}(W) &= \frac{1}{k} \cdot \sum_{1 \leq i \leq k} \left(\sum_{c \in C} \delta(c, \varphi_i) \right) \\ &= \sum_{c \in C} \left(\frac{1}{k} \cdot \sum_{1 \leq i \leq k} \delta(c, \varphi_i) \right) \\ &= \sum_{c \in C} \left(1 - (\frac{1}{2})^{|c|} \right) . \end{aligned}$$

[2] https://doi.org/10.1145/502090.502098

(b) Wegen $|c| \geq 1$ gilt $1 - (\frac{1}{2})^{|c|} \geq \frac{1}{2}$. Demnach gilt nach (a), dass erwartet mindestens die Hälfte aller Klauseln erfüllt ist. Die Einsicht, dass maximal alle Klauseln erfüllt sein können, ergibt die zu zeigende Schranke.

(c) Im Falle $|c| = 3$ gilt $1 - (\frac{1}{2})^{|c|} = \frac{7}{8}$ und damit $\mathbb{E}(W) = \frac{7}{8}m$.

(d)

$$\begin{aligned}\mathbb{E}(W) &= \frac{1}{2}\mathbb{E}(W \mid v_1 \leftarrow \text{wahr}) + \frac{1}{2}\mathbb{E}(W \mid v_1 \leftarrow \text{falsch}) \\ &\leq \frac{1}{2}\mathbb{E}(W \mid v_1 \leftarrow \varphi(v_1)) + \frac{1}{2}\mathbb{E}(W \mid v_1 \leftarrow \varphi(v_1)) \\ &= \mathbb{E}(W \mid v_1 \leftarrow \varphi(v_1))\end{aligned}$$

(e) Betrachte den Fall $\mathbb{E}(W \mid v_1 \leftarrow \text{wahr})$, der Fall $\mathbb{E}(W \mid v_1 \leftarrow \text{falsch})$ ist symmetrisch. Durch $v_1 \leftarrow$ wahr entsteht eine neue Max-Sat Instanz (V', C'), wobei $V' = \{v_2, v_3, \ldots, v_n\}$ und C' aus C hervorgeht, indem Klauseln, die das positive Literal v_1 enthalten, gelöscht werden (weil sie bereits erfüllt sind), und das negative Literal $\neg v_1$ aus allen Klauseln entfernt wird (weil das Literal $\neg v_1$ nicht zur Erfüllung dieser Klauseln beitragen kann). Sei X die Anzahl an Klauseln aus C, die das Literal v_1 enthalten. Sei W' die Anzahl an erfüllten Klauseln in der Instanz (V', C'), deren Erwartungswert sich gemäß (a) in Polynomialzeit ausrechnen lässt. Es gilt $\mathbb{E}(W) = X + \mathbb{E}(W')$.

(f) Gehe induktiv vor. Definiere für $2 \leq i \leq n$ die Wahrheitsbelegung $\varphi(v_i)$ so, dass

$$\begin{aligned}&\mathbb{E}(W \mid v_j \leftarrow \varphi(v_j) \text{ für } 1 \leq j < i \text{ und } v_i \leftarrow \varphi(v_i)) \\ \geq\ &\mathbb{E}(W \mid v_j \leftarrow \varphi(v_j) \text{ für } 1 \leq j < i \text{ und } v_i \leftarrow \neg\varphi(v_i))\,.\end{aligned}$$