Approximationsalgorithmen

Prof. Dr. Klaus Meer, Ameen Naif

Aufgabenblatt 4
Version 15.01.2020

Aufgabe 1.

Sei $\phi = C_1 \wedge ... \wedge C_m$ eine 3-SAT-Formel mit $n \in \mathbb{N}$ vielen Variablen und $m \in \mathbb{N}$ vielen Klauseln so, dass für jedes $y \in \{0,1\}^n$ ein Anteil $\geq \epsilon$ von Klauseln nicht erfüllt ist. Für ein festes $y \in \{0,1\}^n$ führe folgendes Experiment k mal aus:

(a) wähle uniform eine Klausel,

(b) setze y ein und werte aus.

Wie groß muss k mindestens sein, damit bei obigen Experiment die Wahrscheinlichkeit, dass wenigstens eine gewählte Klausel in y nicht erfüllbar ist, mindestens $3\backslash 4$ beträgt.

Aufgabe 2.

Zeigen Sie, dass jede SAT-Instanz mit n vielen Variablen und m vielen Klauseln in polynomieller Zeit auf eine 3-SAT-Instanz reduzierbar ist, wobei $n, m \in \mathbb{N}$. Wie viele Variablen und Klauseln werden dabei in Abhängigkeit von n, m benötigt.

Aufgabe 3.

Beschreiben Sie die Bedingungen für $L \in \text{PCP}(\text{poly(n)}, 0)$ und recherchieren Sie, wie die zugehörige Komplexitätsklasse heißt.

Aufgabe 4.

Beweisen Sie, dass $\text{PCP}(\mathcal{O}(\log \text{n}), 2) = \text{P}$ gilt. Warum ist derselbe Beweis für $\text{PCP}(\mathcal{O}(\log \text{n}), 3) = \text{P}$ nicht erfolgreich?

Aufgabe 5.

Beweisen Sie, dass $\text{PCP}(\mathcal{O}(\log \text{n}), \mathcal{O}(1)) = \text{PCP}(\mathcal{O}(\log \text{n}), \text{poly(n)})$ ist. Dafür darf $\text{3-SAT} \in \text{PCP}(\mathcal{O}(\log \text{n}), \mathcal{O}(1))$ verwendet werden.

Aufgabe 6.

Betrachten wir das *Fractional Knapsack* Problem: Gegeben sein $n \in \mathbb{N}$, die Zahlen $a_1, ..., a_n, c_1, ..., c_n \in \mathbb{N}$ und die Bingröße $B \in \mathbb{N}$. Finde *reelle* Zahlen $x_1, ..., x_n \in [0,1]$ mit

$$\sum_{i=1}^{n} a_i \cdot x_i \leq B \text{ and } \sum_{i=1}^{n} c_i \cdot x_i \text{ ist maximal.}$$

Beweisen Sie, dass das Fractional Knapsack Problem in polynomieller Zeit exakt lösbar ist.

Tip: Ordne die Elemente von S nach c_i/a_i absteigend. Eine Belegung der x_i entsteht, wenn vom ersten Element so viel wie möglich genommen wird, dann vom zweiten in den Restplatz und so weiter. Wie sieht diese Belegung aus? Zeigen Sie, dass sie zulässig und optimal ist.

Aufgabe 7.

Konstruieren Sie Worst-Case Instanzen für den vorgestellten lokalen Such-Algorithmus für MAX-CUT. Zeigen Sie damit, dass die untere Schranke für seine Performance ebenfalls 2 ist.

Aufgabe 8.

In der Lokalen Suche für MAX-CUT soll nun anstelle eines beliebigen Knotens in jedem Schritt der Knoten mit der global besten Verbesserung getauscht werden. Welche untere Schranke für die Performance können Sie für diese Variante beweisen? (Es ginge mit Faktor 2, allerdings mit deutlich komplizierteren Graphen.)