Kombinatorische Optimierung 2, SS 2010

5. Übungsblatt

36. Man betrachte den folgenden Greedy-Algorithmus für das Knapsack-Problem: Man sortiere die Indizes, so dass $\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \ldots \geq \frac{c_n}{w_n}$ und setze $S := \emptyset$. `For` $i = 1$ `to` n `do: If` $\sum_{j \in S \cup \{i\}} w_j \leq W$ `then` $S := S \cup \{i\}$. Zeigen Sie, dass dieser Algorithmus für kein k einen k-Approximationsalgorithmus ergibt.

37. Geben Sie einen exakten Algorithmus für das Knapsack Problem mit Laufzeit $O(nW)$.

38. Sei $c \in \{0, 1, \ldots, k\}^m$ und $s \in [0, 1]^m$. Wie kann man in $O(mk)$-Zeir entscheiden, ob $\max\{cx \colon x \in \mathbb{Z}_+^m, s^t x \leq 1\} \leq k$ gilt? Hier bezeichnet s^t den transponierten Vektor s.

39. Sei $m \in \mathbb{N}$ eine Konstante. Betrachten Sie das folgende Scheduling-Problem: Gegeben seien n Jobs und m Maschinen, Kosten $c_{ij} \in \mathbb{Z}_+$ ($i = 1, 2, \ldots, n$ und $j = 1, 2, \ldots, m$) und Kapazitäten $T_j \in \mathbb{Z}_+$ ($j = 1, 2, \ldots, m$). Bestimmen Sie eine Zuordnung $f \colon \{1, 2, \ldots, n\} \to \{1, 2, \ldots, m\}$ mit $|\{i \in \{1, 2, \ldots, n\} \colon f(i) = j\}| \leq T_j$ für $j = 1, 2, \ldots, m$, welche die Gesamtkosten $\sum_{i=1}^n c_{if(i)}$ minimiert. Man zeige dass dieses Problem ein voll-polynomielles Approximationsschema besitzt.

40. Angenommen, es gilt für eine Instanz $a_1, a_2, \ldots, a_n$ des Bin-Packing-Problems $a_i > \frac{1}{3}$ für alle i. Reduzieren Sie dieses Problem auf das Kardinalitäts-Matching-Problem und zeigen Sie dann, wie man dieses Problem in linearer Zeit löst.

41. Finden Sie eine Instanz I des Bin-Packing-Problems, für welches $FF(I) = 17$ aber $OPT(I) = 10$ (vgl. Vorlesung).

42. Sei k fest. Beschreiben Sie einen pseudopolynomiellen Algorithmus, welcher für eine gegebene Instanz I des Bin-Packing-Problems eine Lösung findet, die höchsten k Behälter benutzt, oder entscheidet, dass keine solche Lösung gibt.

43. Zeigen Sie, dass der First-Fit-Algorithmus und der First-Fit-Decreasing-Algorithmus (vg. Vorlesung) mit $O(n \log n)$ Laufzeit implementiert werden können.

44. Betrachten Sie folgendes Multiprocessor-Scheduling-Problem: Gegeben sei eine Menge A von Jobs, eine positive Zahl $t(a)$ für jedes $a \in A$ (die Ausführungsdauer) und eine Zahl m von Maschinen, jede von denen jeden Job aus A ausführen kann. Bestimmen Sie eine Partition von A in m paarweise disjunkte Mengen A_i, d.h. $A = A_1 \cup A_2 \ldots A_m$, $A_i \cap A_j = \emptyset$, für $i, j = 1, 2, \ldots, n$, $i \neq j$, die $\max_{2 \leq i \leq m} \sum_{a \in A_i} t(a)$ minimiert.

    (a) Zeigen Sie, dass dieses Problem stark NP-schwer ist.

    (b) Zeigen Sie, dass ein Greedy-algorithmus, dass Jobs (in beliebiger Reihenfolge) schrittweise jeweils der zur Zeit am wenigsten benutzten Maschine zuordnet, ein 2-Approximationsalgorithmus ist.

    (c) Zeigen Sie, dass das Problem für jedes festes m ein voll-polynomielles Approximationsschema besitzt.