
Problem Set 1

Approximation Algorithms

WS 2008/09

S. Angelopoulos, N. Megow, and R. Raman

Problem 1 (Dynamic Programming). **1 point**

Find a dynamic programming algorithm that solves the KNAPSACK problem optimally in running time $O(nV)$, where $V = \sum_{i=1}^n v_i$.

Problem 2 (Knapsack). **2 points**

Consider the KNAPSACK problem.

- (i) Show that the algorithm, that greedily picks objects in nonincreasing order of v_i/w_i , does not achieve any bounded performance ratio for the solving the knapsack problem.
- (i) Assume that $w_i \leq B$ for all $i = 1, \dots, n$. Consider the following modification of the greedy algorithm: Let the sorted order (decreasingly, according to the value/weight ratio) be $1, \dots, n$. Find the least k such that $\sum_{i=1}^k w_i > B$, and pick the more profitable of $\{1, \dots, k-1\}$ and $\{k\}$. Show that this modified algorithm has approximation factor of 2.

Hint: Show that $\sum_{i=1}^k v_i$ is an upper bound on the optimum.

- (iii) Give a tight example for the factor 2-approximation algorithm described in (ii).

Problem 3 (Inapproximability). **1 point**

Show that the following problem cannot be approximated within polynomial time computable factor, unless $P=NP$: Given is a graph $G(V, E)$ with *positive* weight function on the edges $w : E \rightarrow \mathbb{R}^+$, and an integer $k \in \mathbb{Z}^+$. Find a subset S of vertices of cardinality k , such that the total weight of edges in the subgraph induced by S is minimized.

Problem 4 (Bin packing). **1 point**

Consider the following very simple algorithm for BIN PACKING: If an item fits into the most recently opened bin, then we put it into this bin. Otherwise, we open a new bin.

- (i) Show that this algorithm is a 2-approximation algorithm.
- (ii) Give a worst case example.

Problem 5 (Scheduling). **2 points**

Consider the following variant of the List Scheduling algorithm discussed in the lecture: order jobs in non-increasing order of processing times before applying list scheduling. Show that this is a $4/3$ -approximation algorithm for scheduling on parallel machines to minimize the makespan.

Hint: Distinguish two cases: $p_j > OPT/3$ and $p_j \leq OPT/3$.