Lehrstuhl für Operations Research
Fakultät für Mathematik
Technische Universität München

Problem Set 2

Greedy Algorithms and Combinatorial Bounds

Approximation Algorithms (MA5517)

Dr. Jannik Matuschke | M.Sc. Marcus Kaiser

This problem set will be discussed in the tutorials on November 6th/7th, 2018.

Problem 2.1 (Knapsack Problem)

In the KNAPSACK problem, we are given as input the size $B \geq 0$ of the knapsack as well as $n \in \mathbb{N}$ items. Each item $i \in [n]$ has a value $v_i \geq 0$ and a size $s_i \geq 0$. The goal is to select a subset of items $I \subseteq [n]$ of maximum total value $v(I)$ which fit in the knapsack, i.e., $s(I) \leq B$. We can assume that $s_i \leq B$ for all $i \in [n]$ and $s([n]) > B$.

Consider the following greedy algorithm for KNAPSACK. Initially, all the items are sorted in order of non-increasing ratio of value to size so that $v_1/s_1 \geq v_2/s_2 \geq \ldots \geq v_n/s_n$. The greedy algorithm puts items in the knapsack in index order until the next item no longer fits, i.e., it finds $k \in [n]$ such that $s([k]) \leq B < s([k+1])$. Finally, it returns either $[k]$ or $\{k+1\}$, whichever has greater value.

i) Prove that this algorithm is a $1/2$-approximation algorithm for KNAPSACK.

ii) Find two (families of) examples: one showing that the last step of the algorithm is essential for the approximation guarantee and another one showing that your analysis of the algorithm is tight.

Problem 2.2 (The Longest Processing Time Rule for Long Processing Times)

Consider the problem of minimizing the makespan on identical parallel machines. Let $m \in \mathbb{N}$ be the number of machines and $n \in \mathbb{N}$ be the number of jobs. Each job $j \in [n]$ has a processing time $p_j \geq 0$. The goal is to find an assignment $\sigma : [n] \to [m]$ which minimizes the makespan $C_{\max} = \max_{i \in [m]} p\big(\sigma^{-1}(i)\big)$.

We consider instances where each processing time is more than one-third the optimal makespan. Further, we can assume that $p_1 \geq p_2 \geq \ldots \geq p_n$ (by possibly relabeling the jobs).

i) For some $C \geq 0$, define $p_j := 0$ for $j \in \{n+1, \ldots, 2m\}$ and the undirected graph

$$G_C := \big([2m], \{\{i,j\} \subseteq [2m] : i \neq j, p_i + p_j \leq C\}\big).$$

Show that there is a schedule with makespan no greater than C if and only if there is a perfect matching in G.

ii) Prove that the longest processing time rule[a] computes an optimal schedule.

Problem 2.3 (Minimum Steiner Tree Problem)

In the MINIMUMSTEINERTREE problem, we are given as input a connected undirected graph $G = (V, E)$ with costs $c_e \geq 0$ for all edges $e \in E$. The set of vertices is partitioned into non-terminals (or Steiner vertices) S and terminals $V \setminus S$. The goal is to find a Steiner tree of minimum cost, i.e., a minimum-cost tree containing all terminals.

i) Suppose that G is complete and the edge costs obey the triangle inequality, i.e., $c_{\{u,w\}} \leq c_{\{u,v\}} + c_{\{v,w\}}$ for all $u, v, w \in V$. Show that computing a minimum spanning tree in the induced graph $G[V \setminus S]$ gives a 2-approximation for such an instance of MINIMUMSTEINERTREE.

ii) Now, consider the general case. For an edge $e = \{v, w\} \in E$, let $\widehat{c}_e$ be the cost of a shortest v-w-path in G w.r.t. c. Consider the metric completion $\widehat{G}$ of G which is the complete graph on V with edge costs $\widehat{c}$. Show how to get a 2-approximation algorithm for MINIMUMSTEINERTREE by applying the algorithm above to $\widehat{G}$.

[a] The longest processing time rule iteratively schedules an unscheduled job with longest processing time on a machine with the currently lowest load

Problem 2.4* (Arc-Disjoint Paths Problem)

In the ArcDisjointPaths problem, we are given as input a directed graph $G = (V, A)$ and $k \in \mathbb{N}$ source-sink pairs $(s_i, t_i) \in V \times V, i \in [k]$. The goal of the problem is to find as many pairwise arc-disjoint source-sink-paths as possible. More formally, we want to find $I \subseteq [k]$ and a s_i-t_i-path P_i for all $i \in I$ such that $|I|$ is as large as possible and for any $i, j \in I, i \neq j$, P_i and P_j are arc-disjoint.

Consider the following greedy algorithm for the problem. Let L be the maximum of $\sqrt{|A|}$ and the diameter of the graph[b]. For i from 1 to k, we check to see if there exists a s_i-t_i path of length at most L in the graph. If there is such a path P_i, we add i to I and remove the arcs of P_i from the graph.

Show that this greedy algorithm is an $\Omega(1/L)$-approximation algorithm for ArcDisjointPaths.

[b]The diameter of a graph $G = (V, A)$ is given by $\max_{v,w \in V} d(v, w)$ where $d(v, w)$ denotes the length of a shortest v-w-path in G.