Technische Universität München
Fakultät für Informatik
Lehrstuhl für Algorithmen und Komplexität
Prof. Dr. Harald Räcke
Richard Stotz

Summer term 2019
Assignment 9
June 24, 2019

Efficient Algorithms and Data Structures II

*Deadline: July 1, 2019, 10:15 am in the **Efficient Algorithms** folder.*

Homework 1 (5 Points)

We are given k stretchable bags $b_1, \ldots, b_k$ and n items $a_1, \ldots, a_n$ with weights $w_1, \ldots, w_n$ and volume $v_1, \ldots, v_n$ respectively, such that $w_i, v_i \leq 1$ and $\sum_{i=1}^{n} w_i = k = \sum_{i=1}^{n} v_i$. We say that a packing of the n items in the k bags is an (α, β)-packing if each bag is filled with weight $\leq \alpha$ and volume $\leq \beta$.
Give an efficient algorithm for obtaining a (3, 3)-packing.

Homework 2 (5 Points)

You have a system that consists of m slow machines and k fast machines. The fast machines can perform *twice* as much work per unit time as the slow machines. You are given a set of n jobs; job i takes time t_i to process on a slow machine and time $t_i/2$ to process on a fast machine. You want to assign each job to a machine so as to minimize the makespan - the makespan is the maximum, over all machines, of the total processing time of jobs assigned to that machine.
Give a polynomial-time algorithm that produces as assignment of jobs to machines with a makespan that is at most three times the optimum.

Homework 3

Given a ground set U of n elements, and a collection $S_1, \ldots, S_\ell \subset U$ and an integer k, we want to pick k sets so as to maximize the number of elements covered.
A greedy algorithm picks the best set in each iteration until k sets are picked.

1. Let x_i be the number of newly covered elements in the ith iteration of the greedy algorithm. Let $z_i = \mathrm{OPT} - \sum_{j=1}^{i} x_j$. Show that $x_{i+1} \geq z_i/k$.

2. Show that $z_{i+1} \leq (1 - 1/k)^{i+1}\,\mathrm{OPT}$.

3. Show that the greedy algorithm achieves an approximation factor of

$$1 - \left(1 - \frac{1}{k}\right)^k > 1 - \frac{1}{e} \ .$$

Tutorial Exercise 1

Consider the following maximization version of the 3-Dimensional Matching Problem. Given disjoint sets X, Y, Z and a set $T \subseteq X \times Y \times Z$ of ordered triples, a subset $M \subseteq T$ is a 3-dimensional matching if each element of $X \cup Y \cup Z$ is contained in at most one of these triples. The Maximum 3-Dimensional Matching Problem is to find a 3-dimensional matching M of maximum cardinality.
Give a polynomial-time algorithm that finds a 3-dimensional matching of size at least 1/3 times the maximum possible size.

Tutorial Exercise 2

Consider an instance of scheduling jobs on identical machines in a scenario where jobs are subject to *precedence constraints*.
We say $i \prec j$ if in any feasible schedule, job i must be completely processed before job j begins processing. A natural variant on the list scheduling algorithm is one in which whenever a machine becomes idle, then any remaining job that is available is assigned to start processing on that machine. A job j is available if all jobs i such that $i \prec j$ have already been completely processed.
Show that this list scheduling algorithm is a 2-approximation algorithm for the problem with precedence constraints.

```
In retrospect ... it is interesting to note that
the original problem that started my research is
still outstanding - namely the problem of planning
or scheduling dynamically over time, particularly
planning dynamically under uncertainty.  If such a
problem could be successfully solved it could eventu-
ally through better planning contribute to the well-
being and stability of the world.
                                      - G. Dantzig
```