Advanced Algorithms 10/04, 2019

Exercise 03

Lecturer: Mohsen Ghaffari *Teaching Assistant: Vaclav Rozhon*

1 Warm-up [Recommended]

1. For the construction of PTAS for the *Bin packing problem* we needed a tricky way of rounding. Here we investigate why this is necessary.

   Construct a set of items with sizes $0 < x_1, \ldots, x_n < 1$ with the following property. While it is possible to pack these into some number m of bins, if we for any $\varepsilon > 0$ consider the set of items with sizes $(1+\varepsilon)x_1, \ldots, (1+\varepsilon)x_n$, we need at least αm bins for some $\alpha > 1$. What is the best α you can get and how does it relate to the issue of designing a PTAS for *Bin packing*?

2. We also faced some issues when we tried to use our solution for the *Bin packing problem* to solve also the *Minimum makespan problem.*

   Construct a set of items with sizes $0 < x_1, \ldots, x_n$ with the following property. While it is possible to construct a solution to *Minimum makespan problem* with m machines of makespan l, any solution with $m-1$ machines will have makespan at least αl for some $\alpha > 1$. What is the best α you can get and how does it relate to the issue of using our *Bin Packing* PTAS algorithm to solve also the *Minimum makespan problem*?

2 Bin Covering (Vazirani 9.7) [Recommended]

Given n items with sizes $a_1, a_2, \ldots, a_n \in [c, 1]$ for some fixed constant $c \in (0, 1)$, give a Polynomial-Time Approximation Scheme (PTAS) for the problem of maximizing the number of bins, subject to the constraint that each bin has items with total size at least 1.

3 Scheduling with Constant Machines (Vazirani 10.2) [Recommended]

Give a Fully Polynomial-Time Approximation Scheme (FPTAS) for the variant of the minimum makespan scheduling problem in which the number m of machines is a fixed constant.

4 Greedy algorithm for minimum makespan

We have seen a greedy algorithm for the *Minimum makespan scheduling problem* with m machines that iterates through the elements $p_1, \ldots, p_n$ in an arbitrary order and in the i-th step assigns the job p_i to the machine with the least load so far. We proved that the algorithm provides 2-approximation guarantee. Prove that it actually provides $2 - 1/m$ approximation guarantee.