Weekplan: Approximation Algorithms I

Inge Li Gørtz

References and Reading

[1] Survival guide for 02282

[2] Algorithm Design, Kleinberg and Tardos, Addison-Wesley, section 11.0, 11.1. *Available on CampusNet.*

[3] The Design of Approximation Algorithms, Williamson and Shmoys, Cambridge Press, section 2.3.

We expect you to read [2] in detail before the lecture. [3] is alternative reading.

In the pretest (see link on Piazza) you can check if you understand the basics before the lecture. The pretest is an absolute minimum for what you should know before the lecture.

Exercises

1 [w] **Piazza** Enroll in the Piazza group for the course.

2 Longest processing time rule Show that LPT obtains an approximation factor of $4/3 - 1/(3m)$.

3 Tight example for LPT Give almost tight examples for the LPT algorithm for scheduling on parallel identical machines. That is, give an example showing that LPT can produce a schedule that is a factor $(4/3 - 1/3m)$ from optimum.

4 Longest processing time rule Prove that for any input where the processing time of each job is more than a third of the optimal makespan, LPT computes an optimal schedule.

5 Shipping consultant[1] You are a consultant for a large Danish shipping company "Ships, Ships, and Ships". They have the following problem. When a ship arrives at a port they have to unload the containers from the ship onto trucks. A ship carries containers with different weights, $w_1, w_2, \ldots, w_n$. Each truck can carry multiple containers, but only up to a total weight of W. The shipping company wants to use as few trucks as possible to unload the ship. This is a NP-complete problem.

You suggest that they use the following greedy algorithm: Consider the containers in any order. Start with an empty truck and begin stacking containers on it until you get to a container that would overload the truck. This truck is now declared loaded and sent away, and you continue with a new truck.

This algorithm might not be optimal, but it is simple and easy to implement in practice.

5.1 Prove that the number of trucks used by the algorithm is within a factor of 2 from the optimum.

5.2 Show that this is tight. That is, give an example, that shows that the algorithm might use (almost) twice as many trucks as the optimum solution.

[1] inspired by [2]

6 Scheduling on related parallel machines (exercise 2.4 in [3]) In this problem, we consider a variant of the problem of scheduling on parallel machines so as to minimize the length of the schedule. Now each machine i has an associated speed s_i, and it takes t_j/s_i units of time to process job j on machine i. Assume that machines are numbered from 1 to m and ordered such that $s_1 \geq s_2 \geq \cdots \geq s_m$. We call these related machines.

6.1 A ρ-relaxed decision procedure for a scheduling problem is an algorithm such that given an instance of the scheduling problem and a deadline D either produces a schedule of length at most $\rho \cdot D$ or correctly states that no schedule of length D is possible for the instance. Show that given a polynomial-time ρ-relaxed decision procedure for the problem of scheduling related machines, one can produce a ρ-approximation algorithm for the problem.

6.2 Consider the following variant of the list scheduling algorithm, now for related machines. Given a deadline D, we label every job j with the slowest machine i such that the job could complete on that machine in time D; that is, $t_j/s_i \leq D$. If there is no such machine for a job j, it is clear that no schedule of length D is possible. If machine i becomes idle at a time D or later, it stops processing. If machine i becomes idle at a time before D, it takes the next job of label i that has not been processed, and starts processing it. If no job of label i is available, it looks for jobs of label $i+1$; if no jobs of label $i+1$ are available, it looks for jobs of label $i+2$, and so on. If no such jobs are available, it stops processing. If not all jobs are processed by this procedure, then the algorithm states that no schedule of length D is possible.

(a) Prove that if the algorithm returns a schedule then it is of length at most $2D$.

(b) Prove that if if the algorithm states that no schedule of length D is possible, then $D < T^*$, where T^* denotes the length/makespan of the optimal schedule.

(c) Prove that the algorithm is a polynomial-time 2-relaxed decision procedure.