

Vorlesung Online Optimierung: Übungen

Sommersemester 2004

Problem 1 (Preemptive single machine scheduling)

Consider the single machine scheduling problem, $1|r_j, \text{pmtn}|\sum w_j C_j$, in which jobs need to be scheduled preemptively on a single machine in such a way that the total weighted completion time is minimized. Each job has a certain processing time, p_j , and a release date r_j . In the online version a job is known to the scheduler only at its release date.

The relaxed version without release dates, $1|\text{pmtn}|\sum w_j C_j$, can be solved (offline) to optimality by Smith's (or the WSPT) rule: schedule jobs in non-increasing value of ratio w_j/p_j . Let $H(j)$ denote the set of jobs that have the same or higher priority than job j , i.e., $H(j) = \{k : w_k/p_k \geq w_j/p_j\}$. Then the optimal value for the problem without release dates can be written as:

$$\text{OPT} = \sum_j w_j \sum_{k \in H(j)} p_k.$$

This value is a lower bound on the optimal value of the original problem (with release dates).

Consider the following two online algorithms.

Algorithmus 1 WSPT

At any time, process the job with highest ratio w_j/p_j among the available (not completed) jobs. Preempt if a job with a higher ratio arrives.

Algorithmus 2 WSRPT

At any time, process the job with highest ratio $w_j/p_j(t)$ among the available (not completed) jobs, where $p_j(t)$ denotes the *remaining* processing time of job j at time t . Preempt if a job with a higher ratio arrives.

- (a) Show that WSPT is 2-competitive.
- (b) Show that WSRPT is 2-competitive.
Hint: How many "lower priority" jobs can be processed between release date of a job and start of this job.

Problem 2 (Preemptive parallel machine scheduling)

Consider the preemptive scheduling problem, $P|r_j, \text{pmtn}|\sum w_j C_j$. n jobs need to be scheduled on m parallel machines. Each job has a processing time p_j , release date r_j and weight w_j . The objective is to minimize the total weighted completion time.

A lower bound on the value of an optimal schedule can be obtained by the optimal value of the single machine scheduling problem where each job has processing time p_j/m (that is, the machine works m times as fast) and there are no release dates.

Algorithmus 3 P-WSPT

At any point in time, schedule the m jobs with highest ratio w_j/p_j among the available, not yet completed, jobs (or fewer if less than m available jobs have not been completed). Preempt if a job arrives with higher ratio w_j/p_j than one of the currently running jobs.

- (a) Show the P-WSPT is 2-competitive.
 Hint: Consider for job j the interval $[r_j, C_j)$ and divide it into two disjoint sets: one in which job j is being processed and one in which job j is not being processed.

Problem 3 (CNN problem)

Consider the following problem, also known as the CNN-problem. Requests arrive one by one, where each request is specified by a point in \mathbb{Z}^2 and need to be served by a server. A request $r_i = (x_i, y_i)$ is served as soon as the x -position of the server equals $x_s = x_i$ or the y -position is $y_s = y_i$. (Consider Manhattan where accidents happen on street crossings and the CNN-reporter only needs to be in the same avenue or lane to be able to film it).

The goal is to serve all requests such that the total distance traveled by the server is minimized.

- (a) Show that the greedy algorithm, that always travels the shortest distance – i.e., in either x or y direction – to serve a request is not (constant) competitive.
 (b) Show that no deterministic algorithm can be better than 2-competitive.
 Hint: requests with $r_i \in \{1, 2\}^2$ suffices.

Problem 4 (Whack-a-Mole problem)

In the Whack-a-Mole problem (WHAM), moles pop up at certain locations and must be whacked by a hammer before they go under ground again. The goal is to whack as many moles as possible. This problem can be viewed as an online problem: Request (moles) arrive over time at points in a metric space and must be served (whacked) before their deadlines (i.e., before disappearing). An online algorithm learns about a request only at its release date.

In the following, we assume that all requests are located on the line, i.e., the position p_i is in $p_i \in \mathbb{R}$, and that the difference between deadline and release date is one time unit.

- (a) Show that no deterministic algorithm can be constant competitive.
 How to deal with the additive constant in the definition of the competitive ratio?
 (b) Show that no randomized algorithm can be constant competitive. You may assume that the additive constant is $b = 0$.

Problem 5 (Paging with lookahead)

Consider the paging problem with cache size k . We give an online algorithm the advantage of lookahead: it knows that request to the next l pages for some constant l .

Show that any deterministic online algorithm that has a lookahead of l pages is at least k -competitive.

Problem 6 (Target Date Assignment – Makespan scheduling)

In the target date assignment problem, requests come in one by one. Each request (job), r_i , is specified by an arrival time t_i and a job length p_i . On arrival, a request needs to be assigned directly to a timeslot in $\{t_i + 1, \dots, t_i + T\}$, where T is some fixed constant $T > 1$.

For a timeslot t , we solve (offline) a scheduling problem on parallel machines where we want to minimize the makespan, i.e., the maximum completion time of a job. The instance of this scheduling problem is given by the requests assigned to this timeslot t .

The objective is to assign the jobs to the timeslot online such that the maximum makespan over all days is minimized.

The algorithm “Two-Opt-First” (TOF), computes on each arrival of a request the (new) optimum, i.e., the optimal value as if the sequence up to this request was the complete sequence. It then assigns a request to the earliest timeslot for which the jobs in the timeslot can be scheduled in such a way that the makespan for this slot is at most twice the optimal value.

(a) Show that TOF is 2-competitive.

Hint: The factor 2 follows immediately from the definition; the only thing to be proven is that the algorithm produces a feasible solution, i.e., all requests can be assigned before their deadlines.