Advanced algorithms

Exercise sheet #7 — Online algorithms

November 16, 2022

Exercice 1 (Online scheduling)**.** Consider m identical machines and a sequence of n tasks of respective duration $p_1, \ldots, p_n$. We must assign each task to a machine, in an online way (the ith task is assigned without knowledge of what comes next, and an assignment is not revocable), aiming at minimizing the termination time of the last task.

Design a greedy algorithm that is $(2 - \frac{1}{m})$-competitive. Recall from the exercise sheet #1 that the optimal scheduling takes a time at least equal to $\max\left(\max_{1 \leqslant i \leqslant n} p_i, \frac{1}{m} \sum_{i=1}^{n} p_i\right)$.

Exercice 2 (Memory management, (pâle 2011))**.** We consider a memory model with a cache, which can hold only one page, and a slower external memory. Accessing the page in cache is free, accessing a page in external memory costs 1. After accessing a page in external memory, we can choose to load it in cache, this costs D. We are interested in a good online strategy to address a stream of requests while minimizing the cost.

We study the following algorithm. Each page p has a counter $c(p)$ that is initially set to 0. Then the memory management performs the following loop.

```
1:  while true do
2:      p ← next requested page
3:      access p (cost ⩽ 1)
4:      c(p) ← c(p) + 1
5:      if c(p) = D then
6:          load p in cache (cost ⩽ D)
7:          while c(p) > 0 do
8:              q ← next requested page
9:              if q ≠ p then
10:                 c(p) ← c(p) − 1
11:                 access q (cost 1)
12:             else
13:                 access q (cost 0)
14:             end if
15:         end while
16:         (end of phase for p)
17:     end if
18: end while
```

We consider that there is no concurrency issue: When there is no request to deal with, the memory management system holds. Notice that in the inner **while** loop, all the counters $c(q)$ for $q \neq p$ are locked.

(a) Consider the case $D = 3$ and the requests $s_1, s_2, \dots$ below. Complete the table with the values taken by the counters along the algorithm.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...
s_i	1	1	4	6	4	1	4	6	1	1	4	1	4	6	1	4	6	6	...
$c(1)$	1	2	2	2	2	3	2												
$c(4)$	0	0	1	1	2	2	2												
$c(6)$	0	0	0	1	1	1	1												

To analyze the performance of the algorithm, let's split the stream $s = (s_1, s_2, \dots)$ into subsets of requests called *phases*, related to each reset of a counter: The phase I_j^p countains the indexes i of the requests s_i received between resets j and $j+1$ of the counter $c(p)$, and such that

- Either s_i increments $c(p)$ (Access to p);
- Or, s_i is received after the request that made $c(p)$ reach the limit D for the $(j+1)$-th time.

In the previous example, we have

$$\begin{aligned} I_0^1 &= (1,2,6,7,8,9,10,11), \quad I_1^1 = (12,\dots), \quad \dots \\ I_0^4 &= (3,5,13,14,15,16,17), \quad \dots \\ I_0^6 &= (4,18,\dots), \quad \dots \end{aligned}$$

Notice that two phases corresponding to distinct pages can be interleaved and that the request I_j^p begins with a sequence of requests to page p and ends with a sequence of at least $D+1$ consecutive requests.

(b) Show that the cost of handling the requests of a single phase is at most $3D$.

Let $\mathcal{A}$ be any other algorithm for handling the given requests.

(c) Given the sequence of requests s, show that we can split the cost paid by $\mathcal{A}$ for each operation (accessing and loading pages) on the different phases such that each phase is assigned a cost at least D.

Hint: Assign the cost paid by $\mathcal{A}$ to load a page that removes q from cache to the phase I_j^q currently open, and analyse the cost paid by $\mathcal{A}$ to handle all other request in each phase.

(d) Deduce the competitive ratio of the algorithm.

Exercice 3 (Online boxing with advice)**.** A sequence σ of n objects, of size $x_1, \dots, x_n \in (0,1]$ is given as input. They must be put into boxes of capacity 1, that is to say, the sum of the sizes of the objects inside a box cannot exceed 1. We aim at minimizing the number of nonempty boxes. We consider here the *online* variant of the problem: when the ith object comes in, it must be given a location immediately and permanently.

The objects are classified into four categories:

- the *tiny* objects, with a size in $(0, \frac{1}{3}]$;
- the *small* objects, with a size in $(\frac{1}{3}, \frac{1}{2}]$;
- the *critical* objects, with a size in $(\frac{1}{2}, \frac{2}{3}]$;
- and the *enormous* objects, with a size in $(\frac{2}{3}, 1]$.

The number m of critical objects is known from the beginning (this is called the advice), and we assume that there are a lot of tiny objects. Consider the following algorithm. The m first boxes are call *critical boxes* and $\frac{2}{3}$ of their capacity is reserved for critical objects. Then the objects are handled depending on their category:

Enormous: the object is put into a new box;
Critical: the object is put into one of the critical boxes;
Small: if possible, the object is put into a box that already contains a small object, otherwise it is put into a new box;
Tiny: if possible, the object is put into the first critical box that can contain the object in its nonreserved third, otherwise it is put into a box that already contains a tiny object, or, as a last possibility, into a new box.

(a) The level of a box is the sum of the sizes of the objects that are in it. At the end of the algorithm, show that the level of boxes that contains enormous or small objects is at least $\frac{2}{3}$ (except maybe for one box).

(b) Assuming that there is a box containing only tiny objects, show that the level of critical boxes (except maybe one) and boxes containing only tiny objects (expect maybe one) is at least $\frac{2}{3}$.

(c) Assume now that no box contains only tiny objects. Assign a weight to objects (depending on the category) so that: in the online algorithm, the boxes (except maybe one) carry a total weight $\geqslant 2$; in any strategy, the boxes contain a weight $\leqslant 3$.

(d) Deduce the competitive ratio of the online algorithm.