OptALI Summer School
Auckland 2011

Online Optimization

Problem 1 (Your own online problem)
Find some online problem in your every day life. Give at least two online algorithms solving that problem.

Problem 2 (Competitive algorithms for maximization problems)
Give a reasonable definition of *competitive ratio* for maximization algorithms. Analogue to the definition for minimization problems, a c-competitive algorithm should be "better" if c is smaller. How should we deal with the additive constant?

Problem 3 (Request answer games)

(a) Formulate the ski rental problem, the ice cream problem as request answer games.

(b) Formulate the scheduling problem from Example 4.3 as request answer problems (using infinite answer sets).

Problem 4 (Bin packing)
In *bin packing*, a finite set of items of size $s_i \in (0, 1]$ is supposed to be packed into bins of unit capacity using the minimum possible number of bins. In online bin packing, an item i has to be packed before the next item $i + 1$ becomes known. Once an item is packed it cannot be removed and put in another bin.

(a) Show that any c-competitive deterministic algorithm for the online Bin Packing Problem has $c \geq 4/3$.

(b) The FIRSTFIT-Algorithm puts an item i always in the first bin that has still enough space to fit in i. If there is no bin left with enough space then a new bin is opened. Show that FIRSTFIT is 2-competitive.

Problem 5 (Potential function and amortised costs)
Recall the application of potential functions and amortized costs in the proof of competitiveness of MTF. Potential functions are a very useful and elegant tool as we want to illustrate with the following examples from the area of data structures.

Consider the data structure D_0 on which n operations can be executed. For $i = 1, \ldots, n$ let c_i denote the cost of an algorithm ALG caused by the ith

operation. Let D_i denote the data structure after the ith operation. A *potential function* Φ assigns a real number $\Phi(D_i)$ to a data structure D_i. The *amortized costs* $\hat{c}_i$ for the ith operation are

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}).$$

The intuition for a potential function and the amortized costs is the following: $\Phi(D_i)$ measures/estimates, how well the current state of the structure is. Imagine $\Phi(D_i)$ as a bank account: if the difference $\Phi(D_i) - \Phi(D_{i-1})$ is negative, then $\hat{c}_i$ underestimates the actual costs c_i. The difference can be balanced by withdrawing the potential-loss/difference from the account.

For the amortized cost holds:

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} \hat{c}_i + \Phi(D_0) - \Phi(D_n). \tag{1}$$

In the following we analyze stack operations. A stack is a Last-in-First-Out memory/buffer S, on which the following operations are defined:

- PUSH(S, x) puts the object x on top of the stack.
- POP(S) returns the topmost object of the stack and removes it. (If the stack is empty then the operation is aborted with an error message.)

Both operations cost $\mathcal{O}(1)$ time units. Additionally we allow the operation MULTIPOP(S, k), which removes the topmost k objects from the stack. This operation takes $\mathcal{O}(k)$ units of time.

(a) Apply the common worst case analysis and show how to obtain a time bound of $\mathcal{O}(n^2)$ for a sequence of n stack operations (beginning with an empty stack). Assume that for the MULTIPOP-operations there are always sufficiently many elements on the stack.

(b) Use the potential $\Phi(S) := |S|$ and amortized costs in order to improve the worst case bound and obtain a time bound of $\mathcal{O}(n)$.

Problem 6

Consider an arbitrary two-person zero-sum game defined by the $n \times m$-Matrix M. Let

$$V_R := \max_i \min_j m_{ij}$$
$$V_C := \min_j \max_i m_{ij}.$$

Show that $v_R \leq v_C$.

Problem 7

Give a two-person zero-sum game such that $V_R < V_C$.

Problem 8 (Cow path problem)
A cow is standing in front of a fence with green yummy grassland behind. You understand that the cow is desperately looking for the hole in the fence. Unfortunately, it does not know which way to go: left or right? What would be the best strategy for the hungry cow in order to find the way through the fence as fast as possible?

This problem can be modeled as the search for an unknown point $a \in \mathbb{R}$ starting from the origin 0. The optimal offline strategy knows the place a of the hole in the fence and, thus, the value of the optimal solution (move straight from the origin to a) is simply the distance $|a|$.

(a) Does there exist a strictly c-competitive algorithm?

(b) Assume that $|a| \geq 1$. Try to construct an 9-competitive algorithm for that problem.

Hint: Consider an algorithm that moves first $\alpha > 1$ units of length to the right, then goes back to the origin, from where it heads α^2 units of length to the left. The ith turning point of that algorithm is $(-1)^{i+1}\alpha^i$.

What can you say about the competitiveness of that algorithm? How would you choose α?

Problem 9 (Online graph matching)
Consider the following online variant of the *matching Problem*. Given is a bipartite graph $G = (H \cup D, R)$, i.e., each directed edge $r \in R$ is of the form $r = (h, d)$ where $h \in H$ and $d \in D$ and $H \cap D = \emptyset$.

Suppose $G = (V, E)$ is an undirected graph. A *matching* is a subset $M \subseteq E$ such that M contains no two incident edges. In a *maximum matching* it is not possible to add an edge without destroying the matching property. A matching M is *perfect* if each node in V is incident to an edge in M.

The dating service *Online-Matching* received data from n men $H = \{h_1, \ldots, h_n\}$ who are interested in a wonderful lady. The service organizes a dance party where these men can find their love. Therefore invitations has been sent to n promising women $D = \{d_1, \ldots, d_n\}$. With the invitation they received an overview of all interested men. Each lady creates a list of her preferred men. At the entrance to the party each girl is assigned a dance partner from her list. If there is no man from a girls list left without a partner, then she is paid some financial compensation and is sent home. Of course, the goal of the dating service is to partner up as many people as possible.

This problem can be modeled as an *online* version of the problem of finding a perfect matching. We are given a bipartite graph $G = (H \cup D, E)$ with $2n$ nodes, and we assume that G has a perfect matching.

An online algorithm knows all men H from the beginning. A request r_i consists of a neighborhood $N(d_i) = \{h \in H\}\,(h, d_i) \in E$ of a woman-node d_i. The online algorithm has to decide upon the arrival of the request to which man in $N(d_i)$ this girl should be assigned to (if possible). The sequence $\sigma = r_1, \ldots, r_n$ consists of a permutation of ladies and the objective is to obtain as many couples as possible.

Consider, now, the following *very simple* algorithm: as soon as a lady arrives she is assign to *any* still single guy who is on her preference list. If there is no guy left then she is sent home.

(a) Prove that this algorithm is 2-competitive.

(Hint: Assume that M is a maximum matching with $|M| < n/2$. Denote by H' the set of men who got a partner through M. Then, $|H'| < n/2$. Make use of the fact that G has a perfect matching.)

(b) Show that *every* deterministic online algorithm (and even each randomized online algorithm against an adaptive offline adversary) has a competitive ratio of no less than 2 for the problem above.

Problem 10 (Bahncard problem)
In Germany, a traveler of the German railway systems, can buy a "Bahncard". It used to cost €60 and is valid for 12 months. Within this period, a traveler can buy train tickets with a reduction of 25%. More generally, the costs of a Bahncard are €B, is valid for a period of T time units and the traveler pays a β fraction of the original cost.

The requests in the Bahncard problem consist of a tuple $r_i = (t_i, p_i)$, where t_i is the time of traveling and p_i is the cost of the (non-reduced) train ticket.

Ofcourse, the traveler does not know when he is going to travel and how much the train tickets he needs will cost beforehand. The question is what is a good strategy for buying a bahncard, or should he never buy a bahncard.

For an algorithm ALG we define the cost of a request r_i as

$$c_{\text{ALG}}(r_i) = \begin{cases} \beta p_i & \text{if ALG has a valid bahncard at time } t_i, \\ p_i & \text{otherwise.} \end{cases}$$

A request is called *reduced*, if ALG has a valid bachcard at time t_i, i.e., it has bought a bahncard in the interval $(t_i - T, t_i]$, otherwise it is called *regular*.

We define

$$c_{\text{CRIT}} = \frac{B}{1-\beta},$$

as the critical cost: the cost for which buying or not buying a bahncard is equally good. For a time interval I, we define $p(I) = \sum_{i:t_i \in I} p_i$, and we say I is *cheap* if $p(I) < c_{\text{CRIT}}$, otherwise we call I *expensive*.

(a) Show that the Ski-rental problem can be formulated as a Bahncard problem.

Hint: Give appropriate B, T, and β.

(b) Show that we can assume w.l.o.g. that

(i) an optimal algorithm nevers buys a bahncard at a time that it still has a valid one, and

(ii) if I is an expensive interval of length at most T, an optimal algorithm has at least one reduced request.

(c) Design an optimal offline algorithm for the Bahncard problem.

(d) Show that a lower bound on the competitive ratio of any deterministic algorithm is $2 - \beta$.

Hint: Consider only requests in one interval of length T, and let the cost of each request be $\varepsilon > 0$.

Problem 11 (Bahncard problem: competitive algorithms)
Consider the Bahncard problem of Exercise 10 and the following two online algorithms.

"Buy never" Never buy a bahncard.

Algorithm sum Buy a bahncard if the costs of all *regular* requests in the last T time units are at least c_{CRIT}.

(a) Show that the algorithm "Buy never" has a competitive ratio of $1/\beta$.

We want to show that SUM is $(2 - \beta)$-competitive. Let $\tau_1, \ldots, \tau_k$ be the time periods in which OPT buys a bahncard. Then, by Exercise 10(a), we know that $[\tau_j, \tau_j + T)$ is an expensive interval, and $[\tau_j + T, \tau_{j+1})$ is a cheap interval.

Consider an expensive interval $I = [\tau_j, \tau_j + T)$. We partition I into three disjoint intervals I_1, I_2, and I_3: in I_1 and I_3 SUM has a valid bahncard, whereas it pays the regular prices during I_2.

(b) Compare the cost of SUM and OPT during a cheap interval.

(c) Compute the costs of OPT during I.

(d) Compute the costs of SUM during I.

Hint: Bound the costs during I_2, using the definition of SUM.

(e) Show that SUM is $(2 - \beta)$ competitive.

Problem 12 (Single machine scheduling: Makespan)
Show that any algorithm that does not introduce unnecessary idle time is 1-competitive for the online problem of minimizing the makespan on a single machine, in both the sequence model, i.e., jobs arrive one by one, and the time stamp model.

Problem 13 (ls for uniformly related machines)
The goal of this exercise is to show an upper bound on the competitive of LS for the problem of scheduling on uniformly related machines so as to minimize the makespan.

Recall that in the setting of uniformly related machines each machine has a speed s_i. We assume that $s_1 \geq \ldots \geq s_m$, and $p_1 \geq \ldots \geq p_n$.

Algorithm ls for uniformly related machine Consider the jobs in the order $1, \ldots, n$. When processing job j, assign this job to a machine on which it completes earliest.

Notice that in the case that all speeds are equal, i.e., the parallel machine case, this definition of LS coincides with the previously given definition.

(a) Show that the following two lower bounds on the value of an optimal schedule hold:

$$\begin{aligned} \text{OPT} &\geq \frac{\sum_j p_j}{\sum_i s_i}, \\ \text{OPT} &\geq \frac{p_j}{s_1}. \end{aligned}$$

(b) Let L_i denote the load of machine M_i and let j be the last job assigned to a machine that has maximal load in the LS-schedule. Show that

$$L_i + \frac{p_j}{s_i} \geq \text{LS}.$$

(c) Show that

$$\text{LS} \leq (1 + \frac{(m-1)s_1}{\sum_i s_i})\text{OPT}.$$

Hint: Rewrite the inequality of exercise (b) and sum appropriately over the machines. Use the second lower bound to bound the value of p_j.

(d) Show that

$$\text{LS} \leq \frac{\sum_i s_i}{s_1}\text{OPT}.$$

Hint: Use the inequality in exercise (b) for an appropriate machine.

(e) Show that LS is $(1 + \sqrt{4m-3})/2$-competitive for scheduling uniformly related machines.

Hint: Set $x = \frac{s_1}{\sum_i s_i}$ and use the bounds on LS in exercises (c) and (d). For which value of x are these bounds maximal?

Problem 14 (Lower bound for ls)
Show that the competitive ratio of $\frac{1+\sqrt{5}}{2}$ of LS for two uniformly related machines is the best possible.

Hint: a sequence of 2 jobs suffices.

Problem 15 (Smith's rule)
Show that Smith's rule computes an optimal schedule for the single machine scheduling problem to minimize total weighted completion time, without release dates ($1 \| \sum w_j C_j$).

Problem 16 (wspt for identical parallel machines)
We consider the problem of minimizing the total weighted completion time on identical parallel machines ($P|r_j, pmtn| \sum w_j C_j$).

A lower bound on the optimal value, can here be obtained by the value of an optimal schedule for the corresponding single machine problem, where each job has processing time p_j/m and there are no release dates.

Algorithm 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 necessary.

Show that P-WSPT is 2-competitive.

Hint: Consider for a job j the interval $[r_j, C_j)$ and divide it into two disjoint intervals: one in which job j is being processed and one in which job j is not being processed. Bound the length of both intervals (and thus C_j) and use the above described lower bound.

Problem 17 (Single machine scheduling: total flow time)
Consider the online problem of scheduling jobs on a single machine in the precence of release dates so as to minimize the total flow time. Recall that the flow time of a job is defined as $F_j = C_j - r_j$.

Show that the SRPT rule, which is optimal for minimizing the total completion time, is also optimal for the total flow time problem.