Prof. Dr. Sándor P. Fekete
Peter Kramer

Exercise notes
22.04.2024

These notes serve as supplementary material to the exercise and cannot replace attendance. If you notice any errors, please email `kramer(at)ibr.cs.tu-bs.de`.

Updated Tuesday 23rd April, 2024, 13:06

Online algorithms have to work with incomplete information. To this end, their *input* is an ordered sequence $\sigma = \sigma_1, \ldots, \sigma_n$ of requests over the input alphabet Σ, the length of which is *not* known in advance. The requests are given to the algorithm one by one, requiring a decision in response before the next request is given. Each response is an element of the output alphabet Γ. An algorithm A can take the input history into account, so we can describe it as a function $A : \Sigma^* \to \Gamma$. Combining the output of A with the input sequence σ in chronological order, we obtain the *run* of A on σ:

$$\rho(A, \sigma) \quad := \quad \sigma_1 A(\sigma_1) \sigma_2 A(\sigma_1 \sigma_2) \ldots \sigma_n A(\sigma_1 \sigma_2 \ldots \sigma_n) \quad \in (\Sigma\, \Gamma)^*$$

To evaluate online algorithms' decisions within a run, a *cost function* $c : (\Sigma\, \Gamma)^* \to \mathbb{R}_{\geq 0}$ is defined over runs of the problem. This function must only be well-defined over valid runs, but can optionally filter by validity, e.g., by assigning infinite cost to invalid runs.

An example. We can formally describe the Ski Rental Problem as follows. Every request corresponds to another day of skiing, and any response corresponds to either renting, purchasing, or previously having purchased a pair of skis.

$$\Sigma = \{\text{ski}\}, \qquad \Gamma = \{\text{rent}, \text{buy}, \text{bought}\}$$

We define the cost function over valid runs as the total money spent by renting at a day rate of R or making a purchase at a one-time cost of B as $c : (\Sigma\, \Gamma)^* \to \mathbb{R}_{\geq 0}$ such that

$$w \mapsto \begin{cases} |w|/2 \cdot R, & \text{if} \quad w \in (\text{ski rent})^* \\ |z|/2 \cdot R + C & \text{if} \quad w = z \text{ ski buy } (\text{ski bought})^*, \quad \text{with} \quad z \in (\text{ski rent})^*. \end{cases}$$

Competitive ratio. Informally, the competitive ratio of an online algorithm A is the worst-case ratio between the cost of an online solution computed by A and that obtained by an optimal offline algorithm OPT's solution over the same input sequence. We formally describe this as the supremum of the ratio over all runs, i.e.,

$$\sup_{\sigma \in \Sigma^*} \left\{ \frac{\text{online solution cost}}{\text{optimal solution cost}} \right\} = \sup_{\sigma \in \Sigma^*} \left\{ \frac{c(\rho(A,\sigma))}{c(\rho(OPT,\sigma))} \right\}.$$

The BahnCard Problem.

We generalize the Ski Rental Problem by considering a scenario in which, rather than buying skis to use for the rest of our skiing career, we can purchase a time-limited discount to save money on future purchases. This is known as the BahnCard Problem, in which we seek to minimize the amount of money we spend on train tickets without exact future knowledge of travel plans. We do this by purchasing a BahnCard, which reduces the cost of any ticket purchase during its validity period by a fixed percentage.

We describe instances of the BahnCard Problem as $BC(C, \beta, T)$, where $C \in \mathbb{R}^+$ is the purchase cost of a BahnCard, $\beta \in \mathbb{R}$ is a discount factor in $(0, 1)$, and $T \in \mathbb{R}^+$ is the BahnCard's validity period. We receive a series of chronologically ordered requests for ticket purchases, all of which must be made. Each request to defines the time at which the purchase is to be made, alongside the base cost of the ticket, i.e., $\sigma_i = (t_i, c_i)$.

In response, an algorithm can either choose to purchase only the ticket, or, in addition, a BahnCard. A BahnCard is valid from the moment it is purchased and can be applied to a ticket that is purchased simultaneously. Buying a BahnCard at time t_i therefore reduces the price of a ticket from c_j to $c_j\beta$, if it is purchased at some time $t_j \in [t_i, t_i + T)$.

We therefore define the input and output alphabets as follows:

$$\Sigma = \mathbb{R}_{\geq 0} \times \mathbb{R}_{>0}, \qquad \Gamma = \{\text{buy } (1), \text{ pass } (0)\}.$$

For each request σ_i, an algorithm A therefore pays based on three cases:

$$c_A(\sigma_i) = \begin{cases} c_i & \text{if a BahnCard is neither owned nor bought,} \\ c_i\beta & \text{if a BahnCard is owned, or} \\ C + c_i\beta & \text{if a BahnCard is bought.} \end{cases}$$

When is buying a BahnCard worth it? Let $I = [b, e)$ be a time interval of length at most T. We define the *base price* $p(\sigma, I)$ as the total cost of tickets requested in I

$$p(\sigma, I) = \sum_{\sigma_i :\, t_i \in I} c_i$$

Buying a BahnCard at the start of I is clearly only efficient if the money saved is more than the initial investment into a BahnCard. We derive the *critical price* c^* to break even with a BahnCard purchased at the start of I as follows:

$$\begin{aligned} & p(\sigma, I) = C + \beta \cdot p(\sigma, I) \\ \Leftrightarrow \quad & (1 - \beta) \cdot p(\sigma, I) = C \\ \Leftrightarrow \quad & p(\sigma, I) = \frac{C}{1 - \beta} \quad = c^*. \end{aligned}$$

Based on this, we say that an interval I is *expensive*, if $p(\sigma, I) \geq c^*$, and *cheap* otherwise. We make the following two observation about the behaviour of any optimal solution.

Lemma 1. *OPT must own a BahnCard at some point in every expensive interval.*

Lemma 2. *OPT will never buy a BahnCard if it already owns a valid one.*

Greedy offline solutions Based on this, is the following an optimal offline algorithm?

```
1: function GREEDY(σ, C, β, T)
2:     for all (t_i, c_i) ∈ σ_1, ..., σ_n do
3:         if GREEDY already owns a BahnCard at t_i then
4:             output pass
5:         else if the interval [t_i, t_i + T) is expensive then
6:             output buy
7:         else
8:             output pass
9:         end if
10:    end for
11: end function
```

No. We can construct a sequence $\sigma = (0, \varepsilon)(T - \varepsilon, c^*)(T + \varepsilon, c^*)(2T - \varepsilon, \varepsilon)$ such that GREEDY purchases two BahnCards, whereas OPT purchases a single one and spends roughly the same amount on tickets:

$$\begin{aligned} c(\mathrm{GRE}, \sigma) &= 2C + 2c^*\beta + 2\varepsilon\beta \\ c(\mathrm{OPT}, \sigma) &= C \ + 2c^*\beta + 2\varepsilon. \end{aligned}$$

For $\varepsilon \to 0$, $\frac{c(\mathrm{GRE},\sigma)}{c(\mathrm{OPT},\sigma)}$ approaches 2, implying that GREEDY is not a suitable approach.

A bound on the competitive ratio of algorithms for the BAHNCARD PROBLEM

Theorem 1. *There is no deterministic online algorithm for the BAHNCARD PROBLEM that has a better (lower) competitive ratio than* $2 - \beta$.

Proof. Let A be such an algorithm. We consider two cases for the behaviour of A.

Case 1: A never buys a BahnCard. In this case, we act as an adversary to maximize the ratio between c_A and c_{OPT} by giving OPT maximal possible savings from a single BahnCard purchase. We therefore send arbitrarily many requests in the time interval $[0, T)$, each of cost $0 < \varepsilon \ll C$. The algorithms incur the following costs.

$$\begin{aligned} c(\mathrm{A}, \sigma) &= \quad p(\sigma, [0, T)) \\ c(\mathrm{OPT}, \sigma) &= C + p(\sigma, [0, T))\beta. \end{aligned}$$

For an increasing number of requests, i.e., $p(\sigma, [0, T)) \to \infty$, $\frac{c(\mathrm{A},\sigma)}{c(\mathrm{OPT},\sigma)}$ approaches $1/\beta$. This is greater than $2 - \beta$ for any value of β, so A is not $2 - \beta$-competitive:

$$\begin{aligned} 1/\beta &\geq 2 - \beta \\ 0 &\geq (2 - \beta)\beta - 1 \\ 0 &\geq -(\beta - 1)^2. \end{aligned}$$

Case 2: A eventually buys a BahnCard. In this case, we again act as an adversary to maximize the ratio between c_A and c_{OPT}. To achieve this, we again send cheap (ε cost) requests in the time interval $[0, T)$ and stop as soon as A purchases a BahnCard.

Let $\sigma = \sigma_1, \ldots, \sigma_n$ be all resulting requests, and $s = \sum_{i \in [1,n-1]} c_i = \varepsilon(n-1)$ the cost of all requests *before* A bought a BahnCard. The algorithms incur the following costs.

$$c(A, \sigma) = s + C + \varepsilon\beta$$

$$c(OPT, \sigma) = \begin{cases} s + \varepsilon & \text{if } s + \varepsilon < c^* \quad \text{(i.e., } [0, T) \text{ is cheap), or} \\ C + \beta(s + \varepsilon) & \text{if } s + \varepsilon \geq c^* \quad \text{(i.e., } [0, T) \text{ is expensive).} \end{cases}$$

We first consider the case that $[0, T)$ is cheap, so $s + \varepsilon < c^*$. Our goal is to maximize the gap between A and OPT with $c(OPT, \sigma) = s + \varepsilon$:

$$\frac{c(A, \sigma)}{c(OPT, \sigma)} = \frac{s + C + \varepsilon\beta}{s + \varepsilon}$$

This ratio shrinks with increasing s, so we assume $s = c^* - \varepsilon$, i.e., the largest possible value for s such that the time interval $[0, T)$ remains cheap:

$$\frac{(c^* - \varepsilon) + C + \varepsilon\beta}{(c^* - \varepsilon) + \varepsilon} = \frac{c^* + C + \varepsilon\beta - \varepsilon}{c^*}$$

$$= \frac{c^* + (1 - \beta)c^* + \varepsilon\beta - \varepsilon}{c^*} = \frac{(2 - \beta)c^* + (\beta - 1)\varepsilon}{c^*}$$

$$\overset{\varepsilon \to 0}{=} \frac{(2 - \beta)c^*}{c^*} = (2 - \beta).$$

It remains to argue the case that $[0, T)$ is expensive, so $s + \varepsilon \geq c^*$. Our goal is to maximize the gap between A and OPT with $c(OPT, \sigma) = C + \beta(s + \varepsilon)$:

$$\frac{c(A, \sigma)}{c(OPT, \sigma)} = \frac{s + C + \varepsilon\beta}{s\beta + C + \varepsilon\beta}$$

This ratio grows with increasing s, so we assume $s = c^* - \varepsilon$, i.e., the smallest possible value for s such that the time interval $[0, T)$ remains expensive:

$$\frac{(c^* - \varepsilon) + C + \varepsilon\beta}{(c^* - \varepsilon)\beta + C + \varepsilon\beta} = \frac{c^* + C + \varepsilon\beta - \varepsilon}{c^*\beta + C}$$

$$\overset{\varepsilon \to 0}{=} \frac{c^* + (1 - \beta)c^*}{c^*\beta + (1 - \beta)c^*} = \frac{(2 - \beta)c^*}{c^*} = (2 - \beta).$$

We conclude that A is $(2 - \beta)$-competitive at best. □