Symposium on Theoretical Aspects of Computer Science 2009 (Freiburg), pp. 385–396
www.stacs-conf.org

ECONOMICAL CACHING

MATTHIAS ENGLERT [1] AND HEIKO RÖGLIN [2] AND JACOB SPÖNEMANN [3] AND BERTHOLD VÖCKING [4]

[1] DIMAP and Department of Computer Science, University of Warwick
E-mail address: englert@dcs.warwick.ac.uk

[2] Department of Computer Science, Boston University
E-mail address: Heiko@Roeglin.org

[3] Institute of Transport Science, RWTH Aachen University
E-mail address: Jacob.Spoenemann@rwth-aachen.de

[4] Department of Computer Science, RWTH Aachen University
E-mail address: voecking@cs.rwth-aachen.de

Abstract. We study the management of buffers and storages in environments with unpredictably varying prices in a competitive analysis. In the economical caching problem, there is a storage with a certain capacity. For each time step, an online algorithm is given a price from the interval $[1, \alpha]$, a consumption, and possibly a buying limit. The online algorithm has to decide the amount to purchase from some commodity, knowing the parameter α but without knowing how the price evolves in the future. The algorithm can purchase at most the buying limit. If it purchases more than the current consumption, then the excess is stored in the storage; otherwise, the gap between consumption and purchase must be taken from the storage. The goal is to minimize the total cost. Interesting applications are, for example, stream caching on mobile devices with different classes of service, battery management in micro hybrid cars, and the efficient purchase of resources.

First we consider the simple but natural class of algorithms that can informally be described as memoryless. We show that these algorithms cannot achieve a competitive ratio below $\sqrt{\alpha}$. Then we present a more sophisticated deterministic algorithm achieving a competitive ratio of

$$\frac{1}{W\left(\frac{1-\alpha}{e\alpha}\right)+1} \in \left[\frac{\sqrt{\alpha}}{\sqrt{2}}, \frac{\sqrt{\alpha}+1}{\sqrt{2}}\right] \ ,$$

where W denotes the Lambert W function. We prove that this algorithm is optimal and that not even randomized online algorithms can achieve a better competitive ratio. On the other hand, we show how to achieve a constant competitive ratio if the storage capacity of the online algorithm exceeds the storage capacity of an optimal offline algorithm by a factor of $\log \alpha$.

1998 ACM Subject Classification: F.1.2, F.2.2.

Key words and phrases: Online Algorithms, Competitive Analysis, Storage Management.

M. Englert is supported by EPSRC grant EP/F043333/1 and DFG grant WE 2842/1. H. Röglin is supported by a fellowship of the German Academic Exchange Service (DAAD) and by DFG through German excellence cluster UMIC. J. Spönemann is supported by DFG Research Training Group 1298 (AlgoSyn). B. Vöcking is supported by DFG through German excellence cluster UMIC.

1. Introduction

In many environments in which resources with unpredictably varying prices are consumed over time, the effective utilization of a storage can decrease the cost significantly. Since decisions have to be made without knowing how the price evolves in the future, storage management can naturally be formulated as an online problem in such environments. In the *economical caching problem* each time step is characterized by a price, a consumption, and a buying limit. In every such time step, an online algorithm has to decide the amount to purchase from some commodity. The algorithm can purchase at most the buying limit. If it purchases more than the current consumption, the excess is stored in a storage of limited capacity; otherwise, the gap between consumption and purchase must be taken from the storage.

This kind of problem does not only arise when purchasing resources like oil or natural gas, but also in other interesting application contexts. Let us illustrate this by two examples, one from the area of mobile communication and one dealing with the energy management in cars. The first example is stream caching on mobile devices with different communication standards like GSM, UMTS, WLAN. Since the price for transmitting data varies between the different standards and since for moving devices it is often unclear which standard will be available in the near future, the problem of cheaply caching a stream can be formulated in our framework. The second example is battery management in micro hybrid cars. In addition to a conventional engine, these cars have an electric motor without driving power that allows the engine to be restarted quickly after it had been turned off during coasting, breaking, or waiting. The power for the electric motor is taken from a battery that must be recharged by the alternator during drive. Since the effectiveness of the conventional engine depends on the current driving situation, the question of when and by how much to recharge the battery can be formulated as an economical caching problem.

Let α denote an upper bound on the price in any step that is known to the online algorithm. Formally, an instance of the economical caching problem is a sequence $\sigma_1\sigma_2\ldots$ in which every step σ_i consists of a price $\beta_i \in [1,\alpha]$, a consumption $v_i \geq 0$, and a buying limit $\ell_i \geq v_i$. During step σ_i, the algorithm has to decide the amount $B_i \in [0,\ell_i]$ to purchase. This amount has to be chosen such that neither the storage load drops below zero nor the storage load exceeds the capacity of the storage, which we can assume to be 1 without loss of generality. Formally, if L_{i-1} denotes the storage load after step σ_{i-1}, then B_i must be chosen such that $L_{i-1} + B_i - v_i \in [0,1]$. The restriction $\ell_i \geq v_i$ is necessary because otherwise covering the consumption might not be possible at all. The *economical caching problem without buying limits* is the special case in which all buying limits are set to infinity.

1.1. Our Results

First we observe that the following simple algorithm achieves a competitive ratio of $\sqrt{\alpha}$ (this also follows as a special case from Theorem 2.7): In every step σ_i with price $\beta_i \leq \sqrt{\alpha}$ buy as much as possible while adhering to the buying limit and the storage capacity. In all other steps buy only as much as necessary to maintain a non-negative storage load.

This algorithm belongs to a more general natural class of algorithms, namely algorithms with *fixed buying functions*. Given an arbitrary buying function $f\colon [1,\alpha] \to [0,1]$, we can define the following algorithm: For every σ_i the amount to purchase is chosen such that the storage load after the step is as close as possible to $f(\beta_i)$ taking into account the buying

limit. For example, the buying function f of the simple algorithm satisfies $f(x) = 1$ for $x \leq \sqrt{\alpha}$ and $f(x) = 0$ for $x > \sqrt{\alpha}$. Informally, algorithms with fixed buying functions can be seen as memoryless and vice versa, in the sense that the action in each step does only depend on the characteristics of that step and the current storage load. However, formally this intuitive view is incorrect since, due to the continuous nature of the problem, an algorithm can encode arbitrary additional information into the storage load. One of our results is a lower bound showing that there is no buying function that gives a better competitive factor than $\sqrt{\alpha}$.

Our main result, however, shows that this is not the best possible competitive factor. We present a more sophisticated deterministic algorithm that achieves a competitive ratio of

$$r := \frac{1}{W\left(\frac{1-\alpha}{e\alpha}\right)+1} \in \left[\frac{\sqrt{\alpha}}{\sqrt{2}}, \frac{\sqrt{\alpha}+1}{\sqrt{2}}\right] \ ,$$

where W denotes the Lambert W function (i.e., the inverse of $f(x) = x \cdot e^x$). We complement this result by a matching lower bound for randomized algorithms, showing that our algorithm is optimal and that randomization does not help. Our lower bounds hold even for the problem without buying limits.

Finally, we consider resource augmentation for the economical caching problem. We show that, for every $z \in \mathbb{N}\backslash\{1\}$, there is a buying function algorithm achieving a competitive ratio of $\sqrt[z]{\alpha}$ against an optimal offline algorithm whose storage capacity is by a factor of $z-1$ smaller than the storage capacity of the online algorithm. In particular, this implies that we obtain a buying function algorithm that is e-competitive against an optimal offline algorithm whose storage capacity is by a factor of $\max\{\lceil \ln(\alpha) \rceil - 1, 1\}$ smaller than the storage capacity of the online algorithm.

1.2. Previous Work

Although the economical caching problem is, in our opinion, a very natural problem with applications from various areas, it seems to have not been studied before in a competitive analysis. However, the problem bears some similarities to the one-way-trading problem introduced by El-Yaniv et al. [4]. In this problem, a trader needs to exchange some initial amount of money in some currency (say, dollars) to some other currency (say, euros). In each step, the trader obtains the current exchange rate and has to decide how much dollars to exchange. However, she cannot exchange euros back to dollars. El-Yaniv et al. present a tight bound of $\Theta(\log \phi)$ on the competitive ratio achievable for the one-way-trading problem, where ϕ denotes the ratio of the worst possible exchange rate and the best possible exchange rate. Results on variations of one- and two-way-trading can also be found in the book by Borodin and El-Yaniv [1] and in a survey by El-Yaniv [3]. In the two-way-trading problem, the trader can buy and sell in both directions. A related problem is portfolio management, which has been extensively studied (see, e.g., [2, 5, 6]).

The special case of the economical caching problem in which consumption occurs only in the last step can be viewed as a one-way-trading problem in which the trader does not start with a fixed amount of dollars but has a fixed target amount of euros. From our proof it is easy to see that our algorithm for the economical caching problem is strictly r-competitive on sequences that are terminated by a step with consumption 1 and price α. Additionally, the sequences used in the lower bound also have the property that they are terminated by such a step. Altogether, this implies that our algorithm is also optimal

for the one-way-trading problem with a fixed target amount and yields a strict competitive ratio of r for that problem.

1.3. Extensions

We can further generalize the economical caching problem. Each step may be characterized by a consumption and a monotonically increasing *price function* $p_i(x)\colon [0,\alpha] \to \mathbb{R}^+$, with $p_i(\alpha) \geq v_i$. The price function has the following meaning: The algorithm can buy up to an amount of $p_i(x)$ at rate at most x. The problem with a single price β_i and a buying limit ℓ_i for each step σ_i is a special case with $p_i(x) = 0$ for $x < \beta_i$ and $p_i(x) = \ell_i$ for $x \geq \beta_i$.

Such price functions appear, for example, implicitly in the stock market. At any given time, all sell orders for a specific stock in the order book define one price function since for every given price x there is a certain number of shares available with an ask price of at most x.

All our results also hold for this more general model. An (online) algorithm can transform an instance for the general problem into an instance of the special problem on the fly: A step with consumption v_i and price function p_i is transformed into a series of steps as follows: First we determine the maximum rate we have to pay to satisfy the demand as $\beta := \inf\{x \mid p_i(x) \geq v_i\}$. Then we generate the following steps (the upper value indicates the price, the middle value the consumption, and the lower value the buying limit)

$$\begin{pmatrix} 1 \\ p_i(1) \\ p_i(1) \end{pmatrix} \begin{pmatrix} 1+\varepsilon \\ p_i(1+\varepsilon) - p_i(1) \\ p_i(1+\varepsilon) - p_i(1) \end{pmatrix} \cdots \begin{pmatrix} \beta-\varepsilon \\ p_i(\beta-\varepsilon) - p_i(\beta-2\varepsilon) \\ p_i(\beta-\varepsilon) - p_i(\beta-2\varepsilon) \end{pmatrix} \begin{pmatrix} \beta \\ p_i(\beta) - p_i(\beta-\varepsilon) \\ p_i(\beta) - p_i(\beta-\varepsilon) \end{pmatrix}$$

for a small ε with $(\beta-1)/\varepsilon \in \mathbb{N}$. Finally, we append the following steps for the remaining prices

$$\begin{pmatrix} \beta+\varepsilon \\ 0 \\ p_i(\beta+\varepsilon) - p_i(\beta) \end{pmatrix} \cdots \begin{pmatrix} \alpha-\varepsilon \\ 0 \\ p_i(\alpha-\varepsilon) - p_i(\alpha-2\varepsilon) \end{pmatrix} \begin{pmatrix} \alpha \\ 0 \\ p_i(\alpha) - p_i(\alpha-\varepsilon) \end{pmatrix}$$

for a small ε with $(\alpha-\beta)/\varepsilon \in \mathbb{N}$.

If ε is small, this transformation does not change the cost of an optimal offline algorithm significantly and hence, our upper bounds on the competitive ratios still hold.

2. Upper Bound

2.1. The Optimal Offline Algorithm

To describe an optimal offline algorithm it is useful to track the cost-profile of the storage contents. For this, we define a monotonically decreasing function $g(x)\colon [0,\alpha] \to [0,1]$ that is initialized with $g(x) := 1$ and changes with each step. In the following, we denote the function $g(x)$ after step σ_i by $g_i(x)$ and the initial function by $g_0(x) = 1$.

The intuition behind $g(x)$ is that, assuming the storage of the optimal offline algorithm is completely filled after step σ_i, a $1 - g(x)$ fraction of the commodity stored in the storage was bought at price x or better.

The change of $g(x)$ from step to step follows two basic rules:

(1) Consumption is satisfied as cheap as possible, i.e., what we remove from the storage is what we bought at the lowest price.
(2) If we have stored something that was bought at a larger than the current price, replace it with commodity bought at the current price. That is, we revoke the decision of the past to buy at the worse price in favor of buying at the current, better price.

Formalizing this yields the following definition

$$g_i(x) := \begin{cases} \min\{g_{i-1}(x) + v_i, 1\} & \text{if } x \le \beta_i, \\ \max\{g_{i-1}(x) + v_i - \ell_i, 0\} & \text{if } x > \beta_i. \end{cases}$$

Using this definition, we can characterize the cost of an optimal offline algorithm. As described above, consumption is satisfied at the best possible price. This gives rise to the cost incurred in step σ_i, namely

$$C_i := \int_0^{\beta_i} \max\{g_{i-1}(x) + v_i - 1, 0\}\, dx \ .$$

Based on these values, we can characterize the cost of an optimal offline algorithm.

Lemma 2.1. *The cost of an optimal offline algorithm is exactly* $\sum_i C_i$.

Due to space limitations, we omit the technical but straightforward proof of this lemma.

2.2. The Optimal Online Algorithm

Our optimal $r := (W\left(\frac{1-\alpha}{e\alpha}\right) + 1)^{-1}$-competitive algorithm is based on the functions $g_i(x)$ introduced in Section 2.1. Note that an online algorithm can compute $g_i(x)$ since the function is solely based on information from the current and past steps. Let the storage level of the online algorithm after step σ_i be denoted by L_i. The initial storage load is $L_0 = 0$. Our algorithm bears some similarity with the following "threat-based" policy for one-way trading defined in [4]: In every step, convert just enough dollars to ensure that the desired competitive ratio would be obtained if in all following steps the exchange rate were equal to the worst possible rate. Our algorithm for the economical caching problem can be described as follows: In every step, the algorithm buys just enough to ensure that it would be strictly r-competitive if after the current step only one more step with consumption 1 and price α occurred that terminated the sequence.

This algorithm can be made explicit as follows: For each step σ_i of the input sequence with price β_i, buying limit ℓ_i, and consumption $v_i \le \ell_i$, the algorithm buys $B_i := v_i + r \cdot \int_1^{\alpha/r} \frac{g_{i-1}(x)-g_i(x)}{\alpha-x}\, dx$ at rate β_i. The storage level after this step is $L_i = L_{i-1} + r \cdot \int_1^{\alpha/r} \frac{g_{i-1}(x)-g_i(x)}{\alpha-x}\, dx$.

Lemma 2.2. *The algorithm above is admissible, that is, it does not buy more than the buying limit and after every step the storage level lies between 0 and 1.*

Proof. In a step σ_i, the algorithm buys

$$B_i = v_i + r \cdot \int_1^{\alpha/r} \frac{g_{i-1}(x) - g_i(x)}{\alpha - x}\, dx \ ,$$

which can be written as

$$\begin{aligned}
&= v_i + r \cdot \int_1^{\beta_i} \frac{g_{i-1}(x) - \min\{g_{i-1}(x) + v_i, 1\}}{\alpha - x}\, dx \\
&\qquad + r \cdot \int_{\beta_i}^{\alpha/r} \frac{g_{i-1}(x) - \max\{g_{i-1}(x) + v_i - \ell_i, 0\}}{\alpha - x}\, dx \\
&\leq v_i + r \cdot \int_{\beta_i}^{\alpha/r} \frac{g_{i-1}(x) - \max\{g_{i-1}(x) + v_i - \ell_i, 0\}}{\alpha - x}\, dx \\
&\leq v_i + r \cdot \int_{\beta_i}^{\alpha/r} \frac{\ell_i - v_i}{\alpha - x}\, dx \leq v_i + r \cdot \int_1^{\alpha/r} \frac{\ell_i - v_i}{\alpha - x}\, dx = \ell_i \ ,
\end{aligned}$$

where the last equation follows from the following observation.

Observation 2.3. For our choice of r,

$$\int_1^{\alpha/r} \frac{1}{\alpha - x}\, dx = \ln(\alpha - 1) - \ln(\alpha - \alpha/r) = \ln\left(1 - \frac{1}{\alpha}\right) - \ln\left(1 - \frac{1}{r}\right) = \frac{1}{r} \ .$$

This observation follows easily from the identity $\ln(-W(x)) = \ln(-x) - W(x)$.

The storage level after step σ_i is

$$\begin{aligned}
L_i &= L_{i-1} + r \cdot \int_1^{\alpha/r} \frac{g_{i-1}(x) - g_i(x)}{\alpha - x}\, dx = L_0 + r \cdot \int_1^{\alpha/r} \frac{g_0(x) - g_i(x)}{\alpha - x}\, dx \\
&= r \cdot \int_1^{\alpha/r} \frac{1 - g_i(x)}{\alpha - x}\, dx = 1 - r \cdot \int_1^{\alpha/r} \frac{g_i(x)}{\alpha - x}\, dx \ ,
\end{aligned}$$

where we use Observation 2.3 to obtain the last equation. This storage level is obviously at most 1. On the other hand,

$$L_i = 1 - r \cdot \int_1^{\alpha/r} \frac{g_i(x)}{\alpha - x}\, dx \geq 1 - r \cdot \int_1^{\alpha/r} \frac{1}{\alpha - x}\, dx = 0 \ ,$$

where the last step follows again from Observation 2.3.

Finally, let us observe that B_i is non-negative. From the definition of g_i it follows that $g_i(x) \leq g_{i-1}(x) + v_i$ for every x. Hence,

$$B_i \geq v_i + r \cdot \int_1^{\alpha/r} \frac{-v_i}{\alpha - x}\, dx = 0 \ ,$$

where the last equality is due to Observation 2.3. ∎

Theorem 2.4. *The algorithm above is $r := (W\left(\frac{1-\alpha}{e\alpha}\right) + 1)^{-1}$-competitive.*

Proof. To prove the theorem we show that, on any sequence, the cost of the algorithm above is at most r times the cost of the optimal offline algorithm plus α. Since α is a constant, this proves the theorem.

We already characterized the cost of an optimal offline algorithm in Section 2.1. The next step in our proof is to bound the C_i's from below. By Lemma 2.1, this yields a lower bound on the cost of an optimal offline algorithm, which is necessary for proving the desired competitive ratio.

Lemma 2.5. *For every step σ_i with $\beta_i \le \alpha/r$,*

$$C_i + \int_0^{\alpha/r} (g_i(x) - g_{i-1}(x))\,dx = \beta_i \cdot v_i - \int_{\beta_i}^{\alpha/r} \min\{\ell_i - v_i, g_{i-1}(x)\}\,dx \ .$$

For every step σ_i with $\beta_i > \alpha/r$,

$$C_i + \int_0^{\alpha/r} (g_i(x) - g_{i-1}(x))\,dx \ge \frac{\alpha}{r} \cdot v_i \ .$$

The only remaining part in the proof is to bound the cost of our algorithm from above. For this, observe that the cost that our algorithm incurs in step σ_i is exactly $\beta_i \cdot B_i$.

Lemma 2.6. *For every step σ_i with $\beta_i \le \alpha/r$,*

$$\beta_i \cdot B_i + \alpha(L_{i-1} - L_i) \le r \left(\beta_i \cdot v_i - \int_{\beta_i}^{\alpha/r} \min\{\ell_i - v_i, g_{i-1}(x)\}\,dx \right) \ .$$

For every step σ_i with $\beta_i > \alpha/r$,

$$\beta_i \cdot B_i + \alpha(L_{i-1} - L_i) \le \alpha \cdot v_i \ .$$

Given the previous lemmas, whose proof will be contained in the full version of this paper, the proof of the theorem follows from elementary calculations: Due to Lemma 2.5 and 2.6,

$$\beta_i \cdot B_i + \alpha(L_{i-1} - L_i) \le r \left(C_i + \int_0^{\alpha/r} (g_i(x) - g_{i-1}(x))\,dx \right) \ ,$$

for every step σ_i. Summing over all steps yields

$$\sum_{i=1}^{n} (\beta_i \cdot B_i) - \alpha(L_n - L_0) \le r \left(\sum_{i=1}^{n} C_i + \int_0^{\alpha/r} (g_n(x) - g_0(x))\,dx \right) \le r \cdot \sum_{i=1}^{n} C_i \ .$$

This concludes the proof of the theorem since the cost of our online algorithm is exactly $\sum_{i=1}^{n}(\beta_i \cdot B_i)$, $\alpha(L_n - L_0) \le \alpha$ and, due to Lemma 2.1, $\sum_{i=1}^{n} C_i$ is equal to the cost of an optimal offline algorithm. ∎

2.3. Algorithm for Larger Storage Capacities

In this section we present a buying function algorithm with a storage capacity of $\lceil \log \alpha / \log c \rceil - 1$ that is c-competitive against an optimal offline algorithm with storage capacity 1. In particular, this implies that for every $z \in \mathbb{N} \setminus \{1\}$, we have an algorithm with storage capacity $z - 1$ that achieves a competitive ratio of $\sqrt[z]{\alpha}$.

Let L_i denote the storage load after step σ_i. Further, we define a buying function

$$B(x) := \max\{\lceil \log \alpha / \log c \rceil - \lfloor \log x / \log c \rfloor - 1, 0\} \ .$$

For each step σ_i of the input sequence with price β_i, buying limit ℓ_i, and consumption $v_i \le \ell_i$, the algorithm buys

$$B_i := \max\{\min\{B(\beta_i) - L_{i-1} + v_i, \ell_i\}, 0\} \ .$$

Hence, the storage load L_i after the i-th step is $L_{i-1} + B_i - v_i$. Again, we have to argue that the algorithm is admissible, i.e., that $0 \le L_i \le \lceil \log \alpha / \log c \rceil - 1$. For $i = 0$ this is obviously the case since $L_0 = 0$. For $i \ge 1$, we observe that

$$\begin{aligned} L_i &= L_{i-1} + B_i - v_i \\ &= L_{i-1} + \max\{\min\{B(\beta_i) - L_{i-1} + v_i, \ell_i\}, 0\} - v_i \\ &= \max\{\min\{B(\beta_i), \ell_i + L_{i-1} - v_i\}, L_{i-1} - v_i\} \ . \end{aligned}$$

Now, on the one hand,

$$\max\{\min\{B(\beta_i), \ell_i + L_{i-1} - v_i\}, L_{i-1} - v_i\} \ge \min\{B(\beta_i), \ell_i + L_{i-1} - v_i\} \ge 0$$

due to the induction hypothesis $L_{i-1} \ge 0$ and since $B(\beta_i) \ge 0$ and $\ell_i \ge v_i$. On the other hand,

$$\max\{\min\{B(\beta_i), \ell_i + L_{i-1} - v_i\}, L_{i-1} - v_i\} \le \max\{B(\beta_i), L_{i-1} - v_i\} \le \lceil \log \alpha / \log c \rceil - 1$$

due to the induction hypothesis $L_{i-1} \le \lceil \log \alpha / \log c \rceil - 1$ and since $B(\beta_i) \le \lceil \log \alpha / \log c \rceil - 1$.

Theorem 2.7. *The above algorithm is c-competitive.*

Proof. To prove the theorem, we use the same functions $g_i(x)$ as in Theorem 2.4. Again, we can characterize the cost of an optimal offline algorithm as $\sum_i C_i$.

In addition, we introduce functions $f_i(x) \colon [0, \alpha] \to [0, \lceil \log \alpha / \log c \rceil - 1]$ defined by $f_0(x) := 0$ and

$$f_i(x) := \begin{cases} \min\{f_{i-1}(x) + B_i, L_i\} = B_i + \min\{f_{i-1}(x), L_{i-1} - v_i\} & \text{if } x \le \beta_i, \\ f_{i-1}(x) & \text{if } x > \beta_i. \end{cases}$$

Clearly, the cost of the online algorithm is equal to $\sum_i \beta_i \cdot B_i$. However, for our proof, we characterize the cost in a different way that is similar to our characterization of the optimal cost. For this, define

$$D_i := \int_0^{\beta_i} \max\{f_{i-1}(x) - L_i + B_i, 0\} \, dx \ .$$

Lemma 2.8. *For every j,*

$$\sum_{i=1}^{j} \beta_i \cdot B_i = \int_0^{\alpha} f_j(x) \, dx + \sum_{i=1}^{j} D_i \ .$$

The goal is to relate D_i to C_i in order to prove the theorem. More precisely, we show that, for every i, $D_i \le c \cdot C_i$. This yields the theorem as

$$\begin{aligned} \sum_{i=1}^{j} \beta_i \cdot B_i &= \int_0^{\alpha} f_j(x) + \sum_{i=1}^{j} D_i \\ &\le \alpha \cdot (\lceil \log \alpha / \log c \rceil - 1) + \sum_{i=1}^{j} D_i \\ &\le \alpha \cdot (\lceil \log \alpha / \log c \rceil - 1) + c \cdot \sum_{i=1}^{j} C_i \ . \end{aligned}$$

In order to show $D_i \le c \cdot C_i$, we need the following invariant.

Lemma 2.9. *For every i and $x \in [0, \alpha/c]$, $L_i - f_i(c \cdot x) - 1 + g_i(x) \geq 0$.*

Using this lemma we obtain

$$\begin{aligned} D_i &= \int_0^{\beta_i} \max\{f_{i-1}(x) - L_i + B_i, 0\}\, dx = \int_0^{\beta_i} \max\{f_{i-1}(x) - L_{i-1} + v_i, 0\}\, dx \\ &\leq \int_0^{\beta_i} \max\{g_{i-1}(x/c) - 1 + v_i, 0\}\, dx = c \cdot \int_0^{\beta_i/c} \max\{g_{i-1}(x) - 1 + v_i, 0\}\, dx \\ &\leq c \cdot \int_0^{\beta_i} \max\{g_{i-1}(x) + v_i - 1, 0\}\, dx = c \cdot C_i \ . \end{aligned}$$

The proofs of Lemmas 2.8 and 2.9 will be contained in the full version of this paper. ■

3. Lower Bounds

3.1. General Lower Bound

Theorem 3.1. *The competitive ratio of any randomized online algorithm for the economical caching problem is at least*

$$r := \frac{1}{W\left(\frac{1-\alpha}{e\alpha}\right) + 1} \ .$$

This also holds for the economical caching problem without buying limits.

Proof. Let A denote an arbitrary randomized online algorithm. For every $\beta \in [1, \alpha/r]$, we construct a sequence Σ_β. This sequence starts with a series Σ'_β of steps without a buying limit, without consumption, and with prices decreasing from α/r to β. To be more precise, let the prices in this series of steps be

$$\frac{\alpha}{r}, \frac{\alpha}{r} - \varepsilon, \frac{\alpha}{r} - 2\varepsilon, \ldots, \beta + \varepsilon, \beta,$$

for a small $\varepsilon > 0$ with $(\alpha/r - \beta)/\varepsilon \in \mathbb{N}$. Since we can choose the discretization parameter ε arbitrarily small, we assume in the following that the prices decrease continuously from α/r to β, to avoid the cumbersome notation caused by discretization. Finally, the sequence Σ_β is obtained by appending one step without a buying limit, consumption 1, and price α to Σ'_β.

Due to the last step with consumption 1 and price α, we can assume that after a sequence of the form Σ_β algorithm A has an empty storage. Otherwise, we can easily modify A such that this property is satisfied without deteriorating its performance. Given this assumption, the behavior of algorithm A on sequences Σ_β can be completely described in terms of a monotonically decreasing *buying function* $f\colon [1, \alpha/r] \to [0, 1]$ with the following meaning: after the subsequence Σ'_β with decreasing prices from α/r to β, the expected storage level of A is $f(\beta)$. Using linearity of expectation, the expected costs of A on Σ_β can be expressed as

$$C_A(\Sigma_\beta) = C_f(\beta) = (1 - f(\beta)) \cdot \alpha + \beta \cdot f(\beta) + \int_\beta^{\alpha/r} f(x)\, dx \ .$$

The first term results from the fact that in the last step of Σ_β algorithm A has to purchase the amount of $1 - f(\beta)$ for price α. The remaining term is illustrated in Figure 1.

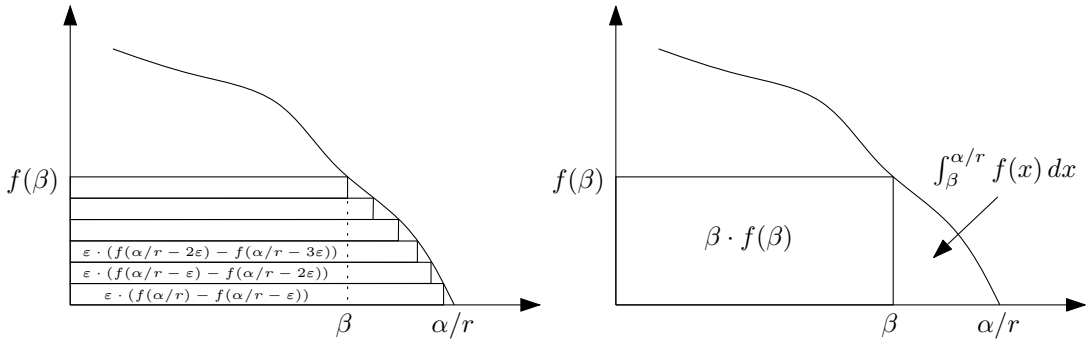

Figure 1: The first figure illustrates the cost of algorithm A on the discrete sequence, and the second figure illustrates its cost on the continuous sequence. Since the function f is monotone, it is integrable on the compact set $[\beta, \alpha/r]$, which in turn implies that for $\varepsilon \to 0$ the costs on the discrete and continuous sequence coincide.

In addition to the actual buying function f of algorithm A, we also consider the buying function g defined by

$$g(x) = r \cdot \left(\ln\left(1 - \frac{x}{\alpha}\right) - \ln\left(1 - \frac{1}{r}\right) \right) \ .$$

This buying function has the property that for all $\beta \in [1, \alpha/r]$

$$C_g(\beta) = (1 - g(\beta)) \cdot \alpha + \beta \cdot g(\beta) + \int_\beta^{\alpha/r} g(x)\,dx = r \cdot \beta \ ,$$

as shown by the following calculation:

$$\begin{aligned}
&(1 - g(\beta))\alpha + \beta \cdot g(\beta) + \int_\beta^{\alpha/r} g(x)\,dx \\
&= (1 - g(\beta))\alpha + \beta \cdot g(\beta) + \int_\beta^{\alpha/r} r \cdot \left(\ln\left(1 - \frac{x}{\alpha}\right) - \ln\left(1 - \frac{1}{r}\right) \right) dx \\
&= (1 - g(\beta))\alpha + \beta \cdot g(\beta) + \left[r \cdot (-\alpha + x) \left(\ln\left(1 - \frac{x}{\alpha}\right) - 1 \right) \right]_\beta^{\alpha/r} - r \cdot \left(\frac{\alpha}{r} - \beta \right) \ln\left(1 - \frac{1}{r}\right) \\
&= (1 - g(\beta))\alpha + \beta \cdot g(\beta) + r \cdot \left((-\alpha + \beta) \ln\left(1 - \frac{1}{r}\right) - (-\alpha + \beta) \ln\left(1 - \frac{\beta}{\alpha}\right) + \left(\beta - \frac{\alpha}{r}\right) \right) \\
&= (1 - g(\beta))\alpha + \beta \cdot g(\beta) + (\alpha - \beta) \cdot g(\beta) + r \cdot \left(\beta - \frac{\alpha}{r}\right) = r \cdot \beta \ .
\end{aligned}$$

Furthermore, g is a valid buying function as it is monotonically decreasing, $g(\alpha/r) = 0$, and

$$g(1) = r \cdot \left(\ln\left(1 - \frac{1}{\alpha}\right) - \ln\left(1 - \frac{1}{r}\right) \right) = 1 \ ,$$

which follows from Observation 2.3.

In order to show the lower bound on A's competitive ratio, we distinguish between two cases: either $f(x) > g(x)$ for all $x \in [1, \alpha/r]$ or there exists an $x \in [1, \alpha/r]$ with $f(x) \leq g(x)$. In the former case, we set $\beta = 1$ and, according to the previous calculations, obtain that $C_A(\Sigma_1) = C_f(1) > C_g(1) = r$. Since the cost of an optimal offline algorithm on Σ_1 is 1, the competitive ratio of algorithm A is bounded from below by r in this case. Now let us consider the case that there exists an $x \in [1, \alpha/r]$ with $f(x) \leq g(x)$. In this case, we set

$$\beta = \sup\{x \in [1, \alpha/r] \mid f(x) \leq g(x)\} \ .$$

Since $f(x) \geq g(x)$ for all $x \geq \beta$, we obtain

$$C_A(\Sigma_\beta) = C_f(\beta) \geq C_g(\beta) = r\beta \ .$$

Combining this with the observation that the cost of an optimal offline algorithm on the sequence Σ_β is β implies that, also in this case, the competitive ratio of A is bounded from below by r.

The argument above shows only that no algorithm can be strictly r'-competitive for $r' < r$ (in fact, it is easy to see that no algorithm can be strictly r'-competitive for $r' < \alpha$). However, the assumption that A has an empty storage after each sequence Σ_β allows us to repeat an arbitrary number of sequences of this kind without affecting the argumentation above, showing that no algorithm can be better than r-competitive. Observe that the buying function of algorithm A can be different in each repetition, which, however, cannot help to obtain a better competitive ratio because β is adopted appropriately in each repetition. ∎

3.2. Lower Bound for Algorithms with Fixed Buying Functions

Theorem 3.2. *The competitive ratio of any randomized online algorithm for the economical caching problem with a fixed buying function is at least $\sqrt{\alpha}$. This also holds for the economical caching problem without buying limits.*

Proof. Let us first consider an algorithm A with an arbitrary but monotonically decreasing buying function f. We will later argue how to extend the proof to functions that are not necessarily monotonically decreasing. We construct a sequence Σ on which A is at least $\sqrt{\alpha}$-competitive as follows: Σ starts with a sequence Σ' that is similar to Σ'_1 from the proof of Theorem 3.1 with the only exception that we decrease the efficiency from α to 1. To be precise, in every step in this sequence there is no consumption, no buying limit, and the prices are

$$\alpha, \alpha - \varepsilon, \alpha - 2\varepsilon, \ldots, 1 + \varepsilon, 1 \ ,$$

for a small ε with $(\alpha - 1)/\varepsilon \in \mathbb{N}$. As in the proof of Theorem 3.1, we simplify the notation by assuming that the price decreases continuously from α to 1. The cost of A on this sequence is

$$q := 1 + \int_1^\alpha f(x)\, dx \ .$$

Let us assume that $f(1) = 1$. Due to the construction of the sequence Σ this can only reduce the cost of A on Σ. We can also assume that $f(\alpha) = 0$ because if A purchases anything at price α, it can easily be seen that A cannot be better than α-competitive.

Now we distinguish between two cases: if $q \geq \sqrt{\alpha}$, then the sequence Σ is formed by appending one step with price α, consumption 1, and no buying limit to Σ'. The cost of an optimal offline algorithm on this sequence is 1, whereas the cost of A is q. Hence, in this case, algorithm A is at least $\sqrt{\alpha}$-competitive.

Now let us assume that $q \leq \sqrt{\alpha}$. After the sequence Σ', the price increases again from 1 to α but this time with consumption. There still is no buying limit and the prices and consumptions are as follows (the upper value indicates the price, the lower value the consumption):

$$\binom{1+\varepsilon}{f(1)-f(1+\varepsilon)}\binom{1+2\varepsilon}{f(1+\varepsilon)-f(1+2\varepsilon)}\cdots\binom{\alpha-\varepsilon}{f(\alpha-2\varepsilon)-f(\alpha-\varepsilon)}\binom{\alpha}{f(\alpha-\varepsilon)} \ .$$

Let us call this sequence Σ''. Observe that consumptions and prices are chosen such that A does not purchase anything during the sequence Σ''. The sequence Σ is formed by appending one step with price α, consumption 1, and no buying limit to $\Sigma'\Sigma''$. On this sequence, the optimal cost is $1+q$: The optimal offline algorithm purchases an amount of 1 in the last step of Σ' for price 1, and then it purchases in every step of Σ'' exactly the consumption. This way the storage is completely filled after the sequence Σ'' and no further cost is incurred in the final step. Similar arguments as in the proof of Theorem 3.1 show that the cost during the sequence Σ'' is q. Since algorithm A does not purchase anything during Σ'', it has to purchase an amount of 1 for the price of α in the final step. Hence, its total cost is $q+\alpha$. For $q \leq \sqrt{\alpha}$, we have

$$\frac{q+\alpha}{q+1} \geq \frac{\sqrt{\alpha}+\alpha}{\sqrt{\alpha}+1} = \sqrt{\alpha} \ .$$

Since $f(\alpha) = 0$, algorithm A has an empty storage after this sequence. Hence, we can repeat this sequence an arbitrary number of times, proving the theorem.

If the buying function f is not monotonically decreasing, we can, for the purpose of this proof, replace f by the monotonically decreasing function $f^*(x) := \sup\{f(y) \mid y \geq x\}$. An algorithm with buying function f behaves the same as an algorithm with buying function f^* on the sequences constructed in this lower bound with respect to f^*. ∎

References

[1] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[2] Thomas M. Cover and Erik Ordentlich. Universal portfolios with side information. *IEEE Transactions on Information Theory*, 42(2):348–363, 1996.

[3] Ran El-Yaniv. Competitive solutions for online financial problems. *ACM Comput. Surv.*, 30(1):28–69, 1998.

[4] Ran El-Yaniv, Amos Fiat, Richard M. Karp, and G. Turpin. Optimal search and one-way trading online algorithms. *Algorithmica*, 30(1):101–139, 2001.

[5] David P. Helmbold, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. On-line portfolio selection using multiplicative updates. In *ICML*, pages 243–251, 1996.

[6] Erik Ordentlich and Thomas M. Cover. On-line portfolio selection. In *COLT*, pages 310–313, 1996.