Ski rental with two general options

Zvi Lotker∗ 
Ben Gurion University

Boaz Patt-Shamir† 
Tel Aviv University

Dror Rawitz‡ 
University of Haifa

June 18, 2008

Abstract

We define and solve a simple extension of the ski-rental problem [4]. In the classical version, the algorithm needs to decide when to switch from renting to buying. In our version, no pure buy option is available: even after switching to the buy option, the algorithm needs to pay some reduced rent. We present an online algorithm for this problem with a matching lower bound.

Keywords: Competitive analysis; Ski rental; Randomized algorithms.

1 Introduction

The ski-rental problem is arguably one of the fundamental questions of online computation. In its original formulation [4], we need to use a resource for an unknown amount of time, and there are two ways to do it: either pay one unit and use the resource from that point onward for any amount of time (the buy option), or pay proportionally to the usage time (the rent option). If the usage time is less than one unit, then the rent option is optimal, and otherwise, the optimal choice is to buy the resource at time 0. It is easy to see that the best deterministic online strategy is to use the rent option for one time unit, and then buy the resource: this ensures competitive ratio 2, i.e., the strategy never pays more than twice the optimum (the worst case is when the game stops immediately after the strategy buys the resource). It is also known that the best randomized online strategy ensures expected competitive ratio of $\frac{e^2}{e-1} \approx 1.58$. (The strategy is to select a time $t \in [0,1]$ at random according to the probability density function $f(t) = \frac{e^t}{e-1}$, and switch from the rent to the buy option at time $t$ if the game has not ended yet.)

In this note we consider a slight generalization of the original problem, where there is no “pure buy”option: there may always be some “rental” to be paid as well. Obviously, real life provides many instances that fall into this category. In this paper we completely characterize this case. Before we state our results, let us define the model more precisely.

∗Department of Communication Systems Engineering, Ben Gurion University, Beer Sheva 84105, Israel. E-mail: zvilo@cse.bgu.ac.il.
†School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: boaz@eng.tau.ac.il.
‡Faculty Science and Science Education & Caesarea Rothschild Institute, University of Haifa, Haifa 31905, Israel. E-mail: rawitz@cri.haifa.ac.il.
Problem statement. There are two options: the first, which we call “slope 1” for a reason which will become apparent shortly, is to pay 1 unit of cost per time unit. In the other option, called “slope 2,” we pay \( a \) units of cost for each time unit of usage, for some real \( 0 \leq a < 1 \). The game begins when the user is at slope 1, and it may switch to slope 2 at any time at cost \((1 - a)\), but no further changes are then allowed. At some time \( t \) the game ends. The total cost to the user at time \( t \) is the time it spent in slope 1, plus \( 1 - a \) if it switched to slope 2, plus \( a \) times the time it spent in slope 2. The task of an algorithm is to determine when to switch to slope 2. We consider randomized algorithms, and we define the cost to the user at time \( t \) to be the expected cost of the algorithm when the stopping time is \( t \). Note that optimal cost is \( t \) if the game stops at time \( t \leq 1 \), or \((1 - a) + at\) if \( t > 1 \). For \( 0 < a < 1 \), the formulation above captures (by change of units) any case where slope 1 cost is described by \( f(x) = a_1x \) and slope 2 cost is \( g(x) = a_2x + b \), assuming non-triviality, i.e., that \( 0 \leq a_2 < a_1 \) and \( b > 0 \).

For \( a = 0 \), this is the basic ski-rental problem. It turns out that this slight generalization is qualitatively different from the classical setting. For example, in classical ski rental, any competitive algorithm must eventually switch to the buy option at some point (otherwise its cost grows without bound while the optimal cost remains constant). When no pure buy is available, the best online algorithm may never switch to the second option.

Geometrical interpretation. The generalized ski-rental problem can be visualized as two intersecting lines on the plane, each corresponding to an option, as shown in Figure 1. The \( x \) axis corresponds to time, and the \( y \) axis corresponds to cost. The rent option corresponds to the steeper line which crosses the \( y \) axis closer to the origin. Our normalized model above assumes that the rent slope crosses the origin, and that the slopes intersect at the point \((1, 1)\).

The main result. The main technical result of this note is the following.

**Theorem 1.1.** Consider the ski rental problem, where in option 1 the buy cost is 0 and the rental rate is 1 for each time unit, and in option 2 the buy cost is \( 1 - a \) and the rental rate is \( a \) for each time unit, where \( 0 < a < 1 \). Then the expected competitive ratio of this instance is exactly \( \frac{e^{1/a}}{e - 1 + a} \).

In Section 2 we prove the upper bound by providing a randomized algorithm and analyzing its
cost. In Section 3 we prove the lower bound by providing an explicit randomized adversary which forces any deterministic online algorithm to pay high expected cost; applying Yao’s Lemma provides the lower bound on any randomized algorithm.

Related work. As mentioned above, the classical ski rental problem, where the buying cost of the first slope, and the rental rate of the second slope, are both 0, was introduced in [4], with optimal strategies achieving competitive factors of 2 (deterministic) and $\frac{e}{e-1}$ (randomized). Karlin et al. [3] apply the randomized strategy to TCP acknowledgment mechanism and other problems. The classical ski rental is sometimes called the leasing problem [1].

Our work is largely motivated by Irani et al. [2] who consider an even more general model, where there may be more than two options. They motivate their work by energy saving: each slope corresponds to some partial “sleep” mode of the system. In [5], the best online randomized algorithm for the latter model is presented. Unlike the current paper, the competitive ratio of the algorithm is not known in general.

2 The Algorithm

Consider a randomized online algorithm for the problem. Let $p_1(t)$ (resp., $p_2(t)$) denote the probability that the algorithm is using slope 1 (resp., slope 2) at time $t$. Note that $p_2(t) = 1 - p_1(t)$ since the algorithm must use either slope 1 or slope 2 at all times $t$. Thus, the algorithm is completely specified by $p_1(t)$.

Below, we develop the algorithm in a way that exposes its rationale. We first find $p_1(t)$ in terms of $c$, where $c$ is the yet-to-be determined competitive factor of the algorithm.

Consider time $t$, for $0 \leq t \leq 1$. The expected rate in which the algorithm spends money (i.e., the derivative of the cost) is

$$1 \cdot p_1(t) + p_2'(t) \cdot (1 - a) + a \cdot p_2(t).$$

The first term is the spending rate due to being at slope 1, the second term is the spending rate due to investment in buying slope 2, and the third term is due to being at slope 2. Recall that if the game stops at time $t < 1$, the optimal strategy is at the first slope, and hence its spending rate is 1. It follows that the algorithm cannot spend at rate larger than $c \cdot 1 = c$. Since we care only about the worst-case competitive ratio, and since spending more allows the algorithm to move faster from slope 1 to slope 2, we let our algorithm spend money at the maximal possible rate, i.e., exactly $c$.

Hence we have

$$c = p_1(t) + p_2'(t) \cdot (1 - a) + a \cdot p_2(t)$$

$$= p_1(t) - p_1'(t) \cdot (1 - a) + a \cdot (1 - p_1(t))$$

$$= a + (1 - a)(p_1(t) - p_1'(t)),$$

since $p_2(t) = 1 - p_1(t)$ and hence $p_2'(t) = -p_1'(t)$. Solving the above differential equation\(^1\) with initial conditions gives

\[^1\text{Recall that if } y'(x) + \alpha y(x) = \beta \text{ for given constants } \alpha \text{ and } \beta, \text{ then } y = \frac{\beta}{\alpha} + \Gamma \cdot e^{-\alpha x}, \text{ where the constant } \Gamma \text{ is usually determined by some boundary condition.}\]
Figure 2: Examples with two slopes: the graphs show $p_1(t)$ (solid lines) and $p_2(t)$ (dashed lines). In the classical case, the second slope is bought with probability 1 by time 1. When $a = 0.5$, there is probability of $\frac{1}{2e-1} \approx 0.23$ of never switching to slope 2.

condition $p_1(0) = 1$ we obtain that

$$p_1(t) = \frac{(1-c)e^t + (c-a)}{1-a}.$$  \hspace{1cm} (1)

In particular, $p_1(1) = \frac{(e-a) - c(e-1)}{1-a}$.

To find the value of the competitive ratio $c$ we consider $t > 1$, where the optimal strategy spends money at rate $a$. We guess that the algorithm stops buying at time 1, i.e., if no switch to slope 2 has occurred by time 1, no switch will ever occur. In other words, we guess that $p_1'(t) = 0$ for $t > 1$. (This guess turns out to be a good guess, as Section 3 proves.) Accepting this guess, we have that for $t > 1$,

$$ca = a + (1-a)p_1(t).$$  \hspace{1cm} (2)

Plugging $p_1(1)$ as found from Eq. (1) into Eq. (2) we can solve for $c$, obtaining that the competitive factor of the algorithm is

$$c = \frac{e}{e - (1-a)},$$

which proves the upper bound of Theorem 1.1. Plugging the value of $c$ back into Eq. (1), we conclude that the algorithm is characterized by

$$p_1(t) = \begin{cases} \frac{a + e - e^t}{e - (1-a)}, & \text{for } 0 \leq t \leq 1 \\ \frac{a}{e - (1-a)}, & \text{for } t > 1. \end{cases}$$

Alternatively, the probability density function of buying slope 2 is \( \frac{d}{dt}p_2(t) = \frac{e^t}{e - (1-a)} \) for $t \in [0, 1]$ and 0 elsewhere.

**Examples.** Figure 2 demonstrates the difference between classical ski rental and the case where there is no “pure buy” option. It contains the optimal randomized online algorithms for two instances: one with $a = 0$ (classical ski rental) and the other with $a = 0.5$. Notice that in the former $p_2(1) = 1$ while in the latter $p_2(1) < 1$. The competitive ratio in the second example is $\frac{e}{e-0.3} \approx 1.225$. 

4
3 Optimality of the algorithm

To show that our algorithm is optimal, we construct a distribution over the inputs and show that the expected cost of any deterministic algorithm is at least \( c \) times the optimal, where \( c = \frac{e}{e-(1-a)} \) is the competitive factor of our algorithm. The distribution is independent of \( a \). Specifically, the probability density function of the stopping at time \( x \) is \( e^{-x} \) for \( 0 \leq x \leq 1 \), and there is a probability of \( 1/e \) of stopping at time 2. The optimal expected cost is therefore

\[
 c_0 \overset{\text{def}}{=} \int_0^1 xe^{-x} \, dx + ((1 - a) + 2a) \cdot \frac{1}{e} = e - \frac{(1-a)}{e}.
\]

(The first term is the cost when the game stops by time 1, and the second term is for the case when the game stops at time 2.)

We now show that any deterministic online algorithm has expected cost at least 1 for the above input distribution. This will allow us to conclude, by Yao’s Lemma (see, e.g., [1]), that no randomized algorithm has competitive ratio better than \( \frac{1}{c_0} = \frac{e}{e-(1-a)} \).

Consider first a deterministic algorithm that moves from slope 1 to slope 2 at time \( t \in [0, 1] \). Then its expected cost is

\[
\int_0^t xe^{-x} \, dx + \int_t^1 (t + (1 - a) + a(x-t))e^{-x} \, dx + (t + (1 - a) + a(2-t)) \cdot \frac{1}{e} =\left(1 - \frac{1 + t}{e^t}\right) + \left(\frac{1 + t}{e^t} - \frac{1 + a + t - at}{e}\right) + \frac{t + 1 + a - at}{e},
\]

which turns out to be identically 1. (The first term is for the case that the algorithm stops by time \( t \), the second term for the case when the algorithm stops between time \( t \) and time 1, and the last is for the case where the algorithm stops at time 2.)

Consider next an algorithm that moves from slope 1 to slope 2 at time \( 1 < t \leq 2 \). Then its expected cost is

\[
\int_0^1 xe^{-x} \, dx + (t + (1 - a) + a(2-t)) \cdot \frac{1}{e} = \frac{e - 2}{e} + \frac{(1 + t)(1 - a) + 2a}{e} > 1,
\]

for any \( t > 1 \) and \( a < 1 \).

Finally, for an algorithm that switches to slope 2 at \( t > 2 \) we have

\[
\int_0^1 xe^{-x} \, dx + 2 \cdot \frac{1}{e} = \frac{e - 2}{e} + \frac{2}{e} = 1,
\]

and the lower bound of Theorem 1.1 is proved as well.

References


