RENT, LEASE OR BUY:
RANDOMIZED ALGORITHMS FOR MULTISLOPE SKI RENTAL

ZVI LOTKER [1], BOAZ PATT-SHAMIR [2], AND DROR RAWITZ [3]

[1] Dept. of Communication Systems Engineering, Ben Gurion University, Beer Sheva 84105 , Israel.

[2] School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel.

[3] Faculty of Science and Science Education & C.R.I., University of Haifa, Haifa 31905, Israel.
E-mail address: `zvilo@cse.bgu.ac.il,boaz@eng.tau.ac.il,rawitz@cri.haifa.ac.il`

Abstract. In the Multislope Ski Rental problem, the user needs a certain resource for some unknown period of time. To use the resource, the user must subscribe to one of several options, each of which consists of a one-time setup cost ("buying price"), and cost proportional to the duration of the usage ("rental rate"). The larger the price, the smaller the rent. The actual usage time is determined by an adversary, and the goal of an algorithm is to minimize the cost by choosing the best option at any point in time. Multislope Ski Rental is a natural generalization of the classical Ski Rental problem (where the only options are pure rent and pure buy), which is one of the fundamental problems of online computation. The Multislope Ski Rental problem is an abstraction of many problems where online decisions cannot be modeled by just two options, e.g., power management in systems which can be shut down in parts. In this paper we study randomized algorithms for Multislope Ski Rental. Our results include the best possible online randomized strategy for any *additive* instance, where the cost of switching from one option to another is the difference in their buying prices; and an algorithm that produces an *e*-competitive randomized strategy for any (non-additive) instance.

1. Introduction

Arguably, the "rent or buy" dilemma is the fundamental problem in online algorithms: intuitively, there is an ongoing game which may end at any moment, and the question is to commit or not to commit. Choosing to commit (the 'buy' option) implies paying large cost immediately, but low overall cost if the game lasts for a long time. Choosing not to commit (the 'rent' option) means high spending rate, but lower overall cost if the game ends quickly. This problem was first abstracted in the "Ski Rental" formulation [10] as follows. In the buy option, a one-time cost is incurred, and thereafter usage is free of charge. In the rent option, the cost is proportional to usage time, and there is no one-time cost. The deterministic solution is straightforward (with competitive factor 2). In the randomized

model, the algorithm chooses a random time to switch from the rent to the buy option (the adversary is assumed to know the algorithm but not the actual outcomes of random experiments). As is well known, the best possible online strategy for classical ski rental has competitive ratio of $\frac{e}{e-1} \approx 1.582$.

In many realistic cases, there may be some intermediate options between the extreme alternatives of pure buy and pure rent: in general, it may be possible to pay only a part of the buying cost and then pay only partial rent. The general problem, called here the *Multislope Ski Rental* problem, can be described as follows. There are several *states* (or *slopes*), where each state i is characterized by two numbers: a *buying cost* b_i and a *rental rate* r_i (see Fig. 1). Without loss of generality, we may assume that for all i, $b_i < b_{i+1}$ and $r_i > r_{i+1}$, namely that after ordering the states in increasing buying costs, the rental rates are decreasing. The basic semantics of the multislope problem is natural: to hold the resource under state i for t time units, the user is charged $b_i + r_i t$ cost units. An adversary gets to choose how long the game will last, and the task is to minimize total cost until the game is over.

The Multislope Ski Rental problem introduces entirely new difficulties when compared to the classical Ski Rental problem. Intuitively, whereas the only question in the classical version is when to buy, in the multislope version we need also to answer the question of what to buy. Another way to see the difficulty is that the number of potential transitions from one slope to another in a strategy is one less than the number of slopes, and finding a single point of transition is qualitatively easier than finding more than one such point.

In addition, the possibility of multiple transitions forces us to define the relation between multiple "buys." Following [2], we distinguish between two natural cases. In the *additive* case, buying costs are cumulative, namely to move from state i to state j we only need to pay the difference in buying prices $b_j - b_i$. In the *non-additive* case, there is an arbitrarily defined transition cost b_{ij} for each pair of states i and j.

Our results. In this paper we analyze randomized strategies for Multislope Ski Rental. (We use the term *strategy* to refer to the procedure that makes online decisions, and the term *algorithm* to refer to the procedure that computes strategies.) Our main focus is the additive case, and our main result is an efficient algorithm that computes the best possible randomized online strategy for any given instance of additive Multislope Ski Rental problem. We first give a simpler algorithm which decomposes a $(k+1)$-slope instance into k two-slope instances, whose competitive factor is $\frac{e}{e-1}$. For the non-additive model, we give a simple e-competitive randomized strategy.

Related Work. Variants of ski rental are implicit in many online problems. The classical (two-slope) ski rental problem, where the buying cost of the first slope and the rental rate of the second slope are 0, was introduced in [10], with optimal strategies achieving competitive factors of 2 (deterministic) and $\frac{e}{e-1}$ (randomized). Karlin et al. [9] apply the randomized strategy to TCP acknowledgment mechanism and other problems. The classical ski rental is sometimes called the *leasing* problem [5].

Azar et al. [3] consider a problem that can be viewed as non-additive multislope ski rental where slopes become available over time, and obtain an online strategy whose competitive ratio is $4 + 2\sqrt{2} \approx 6.83$. Bejerano et al. [4], motivated by rerouting in ATM networks, study the non-additive multislope problem. They give a deterministic 4-competitive strategy, and show that the factor of 4 holds assuming only that the slopes are concave, i.e., when the rent in a slope may decrease with time. Damaschke [6] considers a static version
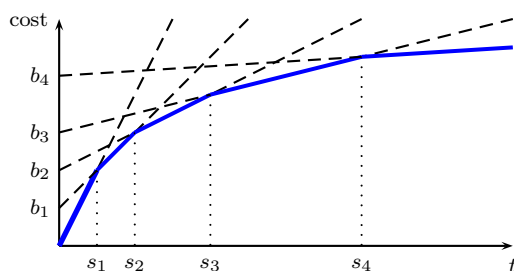
Figure 1: *A multislope ski rental instance with 5 slopes: The thick line indicates the optimal cost as a function of the game duration time.*

of the problem from [3], namely non-additive multislope ski rental problem where each slope is bought "from scratch."[1] For deterministic strategies, [6] gives an upper bound of 4 and a lower bound of $\frac{5+\sqrt{5}}{2} \approx 3.618$; [6] also presents a randomized strategy whose competitive factor is $2/\ln 2 = 2.88$. As far as we know, Damaschke's strategy is the only randomized strategy for multislope ski rental to appear in the literature.

Irani et al. [8] present a deterministic 2-competitive strategy for the additive model that generalizes the strategy for the two slopes case. They motivate their work by energy saving: each slope corresponds to some partial "sleep" mode of the system. Augustine et al. [2] present a dynamic program that computes the best deterministic strategy for non-additive multislope instances. The case where the length of the game is a stochastic variable with known distribution is also considered in both [8, 2].

Meyerson [12] defines the seemingly related "parking permit" problem, where there are k types of permits of different costs, such that each permit allows usage for some duration of time. Meyerson's results indicate that the problems are not very closely related, at least from the competitive analysis point of view: It is shown in [12] that the competitive ratio of the parking permit problem is $\Theta(k)$ and $\Theta(\log k)$ for deterministic and randomized strategies, respectively.

Organization. The remainder of this paper is organized as follows. In Section 2 we define the basic additive model and make a few preliminary observations. In Section 3 we give a simple algorithm to solve the multislope problem, and in Section 4 we present our main result: an optimal online algorithm. An e-competitive algorithm for the non-additive case is presented in Section 5.

2. Problem Statement and Preliminary Observations

In this section we formalize the *additive* version of the multislope ski rental problem. A k-ski rental instance is defined by a set of $k+1$ *states*, and for each state i there is a *buying cost* b_i and a *renting cost* r_i. A state can be represented by a line: the ith state corresponds to the line $y = b_i + r_i x$. Fig. 1 gives a geometrical interpretation of a multislope ski rental instance with five states. We use the terms "state" and "slope" interchangeably.

The requirement of the problem is to specify, for all times t, which slope is chosen at time t. We assume that state transitions can be only forward, and that states cannot be

---

[1]It can be shown that strategies that work for this case also work for the general non-additive case (see Section 5).

skipped, i.e., the only transitions allowed are of the type $i \to i+1$. We stress that this assumption holds without loss of generality in the additive model, where a transition from state $i \to j$ for $j > i+1$ is equivalent to a sequence of transitions $i \to i+1 \to \ldots \to j$ (cf. Section 5). It follows that a *deterministic strategy* for the additive multislope ski rental problem is a monotone non-decreasing sequence $(t_1, \ldots, t_k)$ where $t_i \in [0, \infty)$ corresponds to the transition $i-1 \to i$.

A *randomized strategy* can be described using a probability distribution over the family of deterministic strategies. However, in this paper we use another way to describe randomized strategies. We specify, for all times t, a probability distribution over the set of $k+1$ slopes. The intuition is that this distribution determines the actual cost paid by any online strategy. Formally, a *randomized profile* (or simply a *profile*) is specified by a vector $p(t) = (p_0(t), \ldots, p_k(t))$ of $k+1$ functions, where $p_i(t)$ is the probability to be in state i at time t. The correctness requirement of a profile is $\sum_{i=0}^{k} p_i(t) = 1$ for all $t \geq 0$. Clearly, any strategy is related to some profile. In the sequel we consider a specific type of profiles for which a randomized strategy can be easily obtained.

The performance of a profile is defined by its total accrued cost, which consists of two parts as follows. Given a randomized profile p, the expected *rental cost* of p at time t is

$$R_p(t) \stackrel{\text{def}}{=} \sum_i p_i(t) \cdot r_i \;,$$

and the expected total rental cost up to time t is

$$\int_{z=0}^{t} R_p(z)dz \;.$$

The second part of the cost is the buying cost. In this case it is easier to define the cumulative buying cost. Specifically, the expected *total buying cost* up to time t is

$$B_p(t) \stackrel{\text{def}}{=} \sum_i p_i(t) \cdot b_i \;.$$

The expected *total cost* for p up to time t is

$$X_p(t) \stackrel{\text{def}}{=} B_p(t) + \int_{z=0}^{t} R_p(z)dz \;.$$

The goal of the algorithm is to minimize total cost up to time t for any given $t \geq 0$, with respect to the best possible. Intuitively, we think of a game that may end at any time. For any possible ending time, we compare the total cost of the algorithm with the best possible (offline) cost. To this end, consider the optimal solution of a given instance. If the games ends at time t, the optimal solution is to select the slope with the least cost at time t (the thick line in Fig. 1 denotes the optimal cost for any given t). More formally, the optimal offline cost at time t is

$$\text{OPT}(t) = \min_i (b_i + r_i \cdot t) \;.$$

For $i > 0$, denote by s_i the time t instance where $b_{i-1} + r_{i-1} \cdot t = b_i + r_i \cdot t$, and define $s_0 = 0$. It follows that the optimal slope for a game ending at time t is the slope i for which $t \in [s_i, s_{i+1}]$ (if $t = s_i$ for some i then both slopes $i-1$ and i are optimal).

Finally, let us rule out a few trivial cases. First, note that if there are two slopes such that $b_i \leq b_j$ and $r_i \leq r_j$ then the cost incurred by slope j is never less than the cost incurred slope i, and we may therefore just ignore slope j from the instance. Consequently, we will assume henceforth, without loss of generality, that the states are ordered such that $r_{i-1} > r_i$ and $b_{i-1} < b_i$ for $1 \leq i \leq k$.

Second, using similar reasoning, note that we may consider only strategies that are monotone over time with respect to majorization [11], i.e., strategies such that for any two times $t \leq t'$ we have

$$\sum_{i=0}^{j} p_i(t) \geq \sum_{i=0}^{j} p_i(t') . \tag{2.1}$$

Intuitively, Eq. (2.1) means that there is no point is "rolling back" purchases: if at a given time we have a certain composition of the slopes, then at any later time the composition of slopes may only improve. Note that Eq. (2.1) implies that B_p is monotone increasing and R_p is monotone decreasing, i.e., over time, the strategy invests non-negative amounts in buying, resulting in decreased rental rates.

3. An $\frac{e}{e-1}$-Competitive Algorithm

In this section we describe how to solve the multislope problem by reducing it to the classical two-slope version, resulting in a randomized strategy whose competitive factor is $\frac{e}{e-1}$. This result serves as a warm-up and it also gives us a concrete upper bound on the competitiveness of the algorithm presented in Section 4.

The case of $r_k = 0$. Suppose we are given an instance (b, r) with $k + 1$ slopes, where $r_k = 0$. We define the following k instances of the classical two-slopes ski rental problem: in instance i for $i \in \{1, \ldots, k\}$, we set

instance i: $b_0^i = 0$ and $r_0^i = r_{i-1} - r_i$; $b_1^i = b_i - b_{i-1}$ and $r_1^i = 0$. $\qquad$ (3.1)

Observe that $b_1^i = r_0^i \cdot s_i$, i.e., the two slopes of the ith instance intersect exactly at s_i, their intersection point at the original multislope instance. Now, let $\mathrm{OPT}(t)$ denote the optimal offline solution to the original multislope instance, and let $\mathrm{OPT}^i(t)$ denote the optimal solution of the ith instance at time t, i.e., $\mathrm{OPT}^i(t) = \min\{b_1^i, r_0^i \cdot t\}$. We have the following.

Lemma 3.1. $\mathrm{OPT}(t) = \sum_{i=1}^{k} \mathrm{OPT}^i(t)$.

Proof. Consider a time t and let $i(t)$ be the optimal multislope state at time t. Then,

$$\begin{aligned}
\sum_{i=1}^{k} \mathrm{OPT}^i(t) &= \sum_{i:s_i \leq t} b_1^i + \sum_{i:s_i > t} r_0^i \cdot t \\
&= \sum_{i:s_i \leq t} (b_i - b_{i-1}) + \sum_{i:s_i > t} (r_{i-1} - r_i) \cdot t = b_{i(t)} + r_{i(t)} \cdot t = \mathrm{OPT}(t) .
\end{aligned}$$

$\blacksquare$

Given the decomposition (3.1), it is easy to obtain a strategy for any multislope instance by combining strategies for k classical instances. Specifically, what we do is as follows. Let p^i be the $\frac{e}{e-1}$-competitive profile for the ith (two slope) instance (see [10]). We define a profile $\hat{p}$ for the multislope instance as follows: $\hat{p}_i(t) = p_1^i(t) - p_1^{i+1}(t)$ for $i \in \{1, \ldots, k-1\}$, $\hat{p}_0(t) = p_0^1(t)$, and $\hat{p}_k(t) = p_1^k(t)$. We first prove that the profile is well defined.

Lemma 3.2. *(1)* $p_1^i(t) \leq p_1^{i-1}(t)$ *for every* $i \in \{1, \ldots, k\}$ *and time* t. *(2)* $\sum_{i=0}^{k} \hat{p}_i(t) = 1$.

Proof. By the algorithm for classical ski rental, we have that the strategy for the i instance is $p_1^i(t) = (e^{t \cdot r_0^i/b_1^i} - 1)/(e - 1)$. Claim (1) of the lemma now follows from that fact that $b_1^i/r_0^i = s_i > s_{i-1} = b_1^{i-1}/r_0^{i-1}$ for every $i \in \{1, \ldots, k\}$. Claim (2) follows from the telescopic sum

$$\sum_{i=0}^{k} \hat{p}_i(t) = p_0^1(t) + \sum_{i=1}^{k-1}(p_1^i(t) - p_1^{i+1}(t)) + p_1^k(t) = p_0^1(t) + p_1^1(t) = 1 .$$

∎

Next, we show how to convert the profile $\hat{p}$ into a strategy. Note that the strategy uses a single random experiment, since arbitrary dependence between the different p_is are allowed.

Lemma 3.3. *Given $\hat{p}$ one can obtain an online strategy whose profile is $\hat{p}$.*

Proof. Define $\hat{P}_i(t) \stackrel{\text{def}}{=} \sum_{j \geq i} \hat{p}_j(t)$ and let U be a random variable that is chosen uniformly from $[0, 1]$. The strategy is as follows: we move from state i to state $i + 1$ when $U = \hat{P}_i(t)$ for every state i. Namely, the ith transition time t_i is the time t such that $U = \hat{P}_i(t)$. ∎

Thus we obtain the following:

Theorem 3.4. *The expected cost of the strategy defined by $\hat{p}$ is at most $\frac{e}{e-1}$ times the optimal offline cost.*

Proof. We first show that by linearity, the expected cost to the combined strategy is the sum of the costs to the two-slope strategies, i.e., that $X_{\hat{p}}(t) = \sum_{i=1}^{k} X_{p^i}(t)$. For example, the buying cost is

$$B_{\hat{p}}(t) = \sum_{i=0}^{k} \hat{p}_i(t) \cdot b_i = \sum_{i=0}^{k-1}(p_1^i(t) - p_1^{i+1}(t)) \cdot b_i + p_1^k(t) \cdot b_k = \sum_{i=1}^{k} p_1^i(t) \cdot (b_i - b_{i-1}) = \sum_{i=1}^{k} B_{p^i}(t) .$$

Similarly, $R_{\hat{p}}(t) = \sum_{i=1}^{k} R_{p^i}(t)$ by linearity, and therefore,

$$X_{\hat{p}}(t) = B_{\hat{p}}(t) + \int_{z=0}^{t} R_{\hat{p}}(z)dz = \sum_{i=1}^{k} B_{p^i}(t) + \int_{z=0}^{t}\left(\sum_{i=1}^{k} R_{p^i}(z)\right)dz = \sum_{i=1}^{k} X_{p^i}(t) .$$

Finally, by Lemma 3.1 and the fact that the strategies $p^1, \ldots, p^k$ are $\frac{e}{e-1}$-competitive we conclude that

$$X_{\hat{p}}(t) = \sum_{i=1}^{k} X_{p^i}(t) \leq \sum_{i=1}^{k} \frac{e}{e-1} \cdot \text{OPT}^i(t) = \frac{e}{e-1} \cdot \text{OPT}(t)$$

which means that $\hat{p}$ is $\frac{e}{e-1}$-competitive. ∎

The case of $r_k > 0$. We note that if the smallest rental rate r_k is positive, then the competitive ratio is strictly less that $\frac{e}{e-1}$: this can be seen by considering a new instance where r_k is subtracted from all rental rates, i.e., $b_i' = b_i$ and $r_i' = r_i - r_k$ for all $0 \leq i \leq k$. Suppose p is $\frac{e}{e-1}$-competitive with respect to (r', k') (note that $r_k' = 0$). Then the competitive ratio of p at time t w.r.t. the original instance is:

$$c(t) = \frac{X_p(t)}{\text{OPT}(t)} = \frac{X_p'(t) + r_k \cdot t}{\text{OPT}'(t) + r_k \cdot t} \leq \frac{\frac{e}{e-1} \cdot \text{OPT}'(t) + r_k \cdot t}{\text{OPT}'(t) + r_k \cdot t} = \frac{e}{e-1} - \frac{1}{e-1} \cdot \frac{1}{\frac{\text{OPT}'(t)}{r_k \cdot t} + 1}$$

$\frac{d}{dt}\text{OPT}'(t) = r_i - r_k$ for $t \in [s_{i-1}, s_i)$. Hence, the ratio $\frac{\text{OPT}'(t)}{r_k \cdot t}$ is monotone decreasing, and thus $c(t)$ is monotone decreasing as well. It follows that

$$c \leq \frac{e}{e-1} - \frac{1}{e-1} \cdot \frac{1}{\frac{r_0 - r_k}{r_k} + 1} = \frac{e - r_k/r_0}{e-1}$$

Observe that $c = \frac{e}{e-1}$ when $r_k = 0$, and that $c = 1$ when $r_k = r_0$ (i.e., when $k = 0$).

4. An Optimal Online Algorithm

In this section we develop an optimal online strategy for any given additive multislope ski rental instance. We reduce the set of all possible strategies to a subset of much simpler strategies, which on one hand contains an optimal strategy, and on the other hand is easier to analyze, and in particular, allows us to effectively find such an optimal strategy.

Consider an arbitrary profile. (Recall that we assume w.l.o.g. that no slope is completely dominated by another.) As a first simplification, we confine ourselves to profiles where each p_i has only finitely many discontinuities. This allows us to avoid measure-theoretic pathologies without ruling out any reasonable solution within the Church-Turing computational model. It can be shown that we may consider only continuous profiles (details omitted).

So let such a profile $p = (p_0, \ldots, p_k)$ be given. We show that it can be transformed into a profile of a certain structure without increasing the competitive factor. Our chain of transformations is as follows. First, we show that it suffices to consider only simple profiles we call "prudent." Prudent strategies buy slopes in order, one by one, without skipping and without buying more than one slope at a time. We then define the concept of "tight" profiles, which are prudent profiles that spend money at a fixed rate relative to the optimal offline strategy. We prove that there exists a tight optimal profile. Furthermore, the best tight profile can be effectively computed: Given a constant c, we show how to check whether there exists a tight c-competitive strategy, and this way, using binary search on c, we can find the best tight strategy. Finally, we explain how to construct that profile and a corresponding strategy.

4.1. Prudent and Tight Profiles

Our main simplification step is to show that it is sufficient to consider only profiles that buy slopes consecutively one by one. Formally, *prudent* profiles are defined as follows.

Definition 4.1 (active slopes, prudent profiles)**.** A slope i is *active* at time t if $p_i(t) > 0$. A profile is called *prudent* if at all times there is either one or two consecutive active slopes.

At any given time t, at least one slope is active because $\sum_i p_i(t) = 1$ by the problem definition. Considering Eq. (2.1) as well, we see that a continuous prudent profile progresses from one slope to next without skipping any slope in between: once slope i is fully "paid for" (i.e., $p_i(t) = 1$), the algorithm will start buying slope $i + 1$.

We now prove that the set of continuous prudent profiles contains an optimal profile. Intuitively, the idea is that a non-prudent profile must have two non-consecutive slopes with positive probability at some time. In this case we can "shift" some probability toward a middle slope and only improve the overall cost.

Theorem 4.2. *If there exists a continuous c-competitive profile p for some $c \geq 1$, then there exists a prudent c-competitive profile $\tilde{p}$.*

Proof. Let $p = (p_0, \ldots, p_k)$ be a profile and suppose that all the p_is are continuous. It follows that B_p is also continuous. Define $\mathrm{best}(t) = \max\{i : b_i \leq B_p(t)\}$ and $\mathrm{next}(t) = \min\{i : b_i \geq B_p(t)\}$. In words, $\mathrm{best}(t)$ is the most expensive slope that is fully within the buying budget of p at time t, and $\mathrm{next}(t)$ is the most expensive slope that is at least partially within the buying budget of p at time t. Obviously, $\mathrm{best}(t) \leq \mathrm{next}(t) \leq \mathrm{best}(t) + 1$ for all t. Now, we define $\tilde{p}$ as follows:

$$
\tilde{p}_i(t) = \begin{cases} \dfrac{b_{\mathrm{next}} - B_p(t)}{b_{\mathrm{next}} - b_{\mathrm{best}}} & i = \mathrm{best}(t) \text{ and } \mathrm{best}(t) \neq \mathrm{next}(t), \\[2mm] \dfrac{B_p(t) - b_{\mathrm{best}}}{b_{\mathrm{next}} - b_{\mathrm{best}}} & i = \mathrm{next}(t) \text{ and } \mathrm{best}(t) \neq \mathrm{next}(t), \\[2mm] 1 & i = \mathrm{best}(t) = \mathrm{next}(t), \\[2mm] 0 & \text{otherwise.} \end{cases}
$$

It is not hard to verify that $\sum_i p_i(t) = 1$ for every time t. Furthermore, observe that $\tilde{p}$ is prudent, because B_p is continuous. It remains to show that $\tilde{p}$ is c-competitive. We do so by proving that $B_{\tilde{p}}(t) = B_p(t)$ and $R_{\tilde{p}}(t) \leq R_p(t)$ for all t. First, directly from definitions we have

$$
\begin{aligned}
B_{\tilde{p}}(t) &= p_{\mathrm{best}(t)}(t) \cdot b_{\mathrm{best}(t)} + p_{\mathrm{next}(t)}(t) \cdot b_{\mathrm{next}(t)} \\[2mm]
&= \frac{b_{\mathrm{next}(t)} - B_p(t)}{b_{\mathrm{next}(t)} - b_{\mathrm{best}(t)}} \cdot b_{\mathrm{best}(t)} + \frac{B_p(t) - b_{\mathrm{best}(t)}}{b_{\mathrm{next}(t)} - b_{\mathrm{best}(t)}} \cdot b_{\mathrm{next}(t)} = B_p(t) \ .
\end{aligned}
$$

Consider now rental payments. To prove that $R_{\tilde{p}}(t) \leq R_p(t)$ for every time t we construct inductively a sequence of probability distributions $p = p^0, \ldots, p^\ell = \tilde{p}$. The first distribution p^0 is defined to be p. Suppose now that p^j is not prudent. Distribution p^{j+1} is obtained from p^j as follows. For any t such that there are two non-consecutive slopes with positive probability, let $i_1(t), i_2(t), i_3(t)$ be any three slopes such that $i_1(t) = \mathrm{argmin}\{i : p_i^j(t) > 0\}$, $i_3(t) = \mathrm{argmax}\{i : p_i^j(t) > 0\}$, and $i_1(t) < i_2(t) < i_3(t)$ (such $i_2(t)$ exists because p^j is not prudent). Define

$$
p_i^{j+1}(t) = \begin{cases} p_i^j(t) - \dfrac{\Delta^j(t)}{b_{i_2(t)} - b_{i_1(t)}} & i = i_1(t), \\[3mm] p_i^j(t) + \dfrac{\Delta^j(t)}{b_{i_2(t)} - b_{i_1(t)}} + \dfrac{\Delta^j(t)}{b_{i_3(t)} - b_{i_2(t)}} & i = i_2(t), \\[3mm] p_i^j(t) - \dfrac{\Delta^j(t)}{b_{i_3(t)} - b_{i_2(t)}} & i = i_3(t), \\[3mm] p_i^j(t) & i \notin \{i_1(t), i_2(t), i_3(t)\} \end{cases}
$$

where $\Delta^j(t) > 0$ is maximized so that $p_i^{j+1}(t) \geq 0$ for all i. Intuitively, we shift a maximal amount of probability mass from slopes $i_1(t)$ and $i_3(t)$ to the middle slope $i_2(t)$. The fact that $\Delta^j(t)$ is maximized means that we have either that $p_{i_1}^{j+1}(t) = 0$, or $p_{i_3}^{j+1}(t) = 0$, or both. In any case, we may already conclude that $\ell < k$. Also note that by construction, for all t we have $B_{p^{j+1}}(t) = \sum_i p_i^{j+1}(t) \cdot b_i = \sum_i p_i^j(t) \cdot b_i = B_{p^j}(t)$. Hence, $p^\ell = \tilde{p}$.

As to the rental cost, fix a time t, and consider now the rent paid by p^j and p^{j+1}:

$$
\begin{aligned}
R_{p^j}(t) - R_{p^{j+1}}(t) &= \\[2mm]
&= r_{i_1(t)} \frac{\Delta^j(t)}{b_{i_2(t)} - b_{i_1(t)}} - r_{i_2(t)} \left(\frac{\Delta^j(t)}{b_{i_2(t)} - b_{i_1(t)}} \frac{\Delta^j(t)}{b_{i_3(t)} - b_{i_2(t)}} \right) + r_{i_3(t)} \frac{\Delta^j(t)}{b_{i_3(t)} - b_{i_2(t)}} \\[3mm]
&= \Delta^j(t) \cdot \left(\frac{r_{i_1(t)} - r_{i_2(t)}}{b_{i_2(t)} - b_{i_1(t)}} - \frac{r_{i_2(t)} - r_{i_3(t)}}{b_{i_3(t)} - b_{i_2(t)}} \right) > 0
\end{aligned}
$$

where the last inequality follows from the fact that if $i < j$, then $\frac{b_j - b_i}{r_i - r_j}$ is the x coordinate of the intersection point between the slopes i and j. ∎

Our next step is to consider profiles that invest in buying as much as possible under some spending rate cap. Our approach is motivated by the following intuitive observation.

Observation 4.3. Let p^1 and p^2 be two randomized prudent profiles. If $B_{p^1}(t) \geq B_{p^2}(t)$ for every t, then $R_{p^1}(t) \leq R_{p^2}(t)$ for every t.

In other words, investing available funds in buying as soon as possible results in lower rent, and therefore in more available funds. Hence, we define a class of profiles which spend money as soon as possible in buying, as long as there is a better slope to buy, namely as long as $p_k(t) < 1$.

Definition 4.4. Let $c \geq 1$. A prudent c-competitive profile p is called *tight* if $X_p(t) = c \cdot \text{OPT}(t)$ for all t with $p_k(t) < 1$.

Clearly, if the last slope is flat, i.e., $r_k = 0$, then it must be the case that $p_k(s_k) = 1$ for any profile with finite competitive factor: otherwise, the cost to the profile will grow without bound while the optimal cost remains constant. However, it is important to note that if $r_k > 0$, there may exist an optimal profile p that never buys the last slope, but still its expected spending rate as t tends to infinity is $c \cdot r_k$.

It is easy to see that a tight profile can achieve any achievable competitive factor.

Lemma 4.5. *If there exists a c-competitive prudent profile p for some $c \geq 1$, then there exists a c-competitive tight profile $\tilde{p}$.*

Proof. Let $\tilde{p}$ be the prudent profile satisfying $X_{\tilde{p}}(t) = c \cdot \text{OPT}(t)$ for all t for which $\tilde{p}_k(t) < 1$. We need to show that $\tilde{p}$ is feasible. Since by definition, p buys with any amount left, it suffices to show that for all t, the rent paid by p is at most $c \cdot \frac{d}{dt}\text{OPT}(t)$. Indeed, $R_{\tilde{p}}(t) \leq R_p(t)$ for every t due to Observation 4.3, and since p is c-competitive it follows that $R_p(t) \leq c \cdot \frac{d}{dt}\text{OPT}(t)$ and we are done. ∎

4.2. Constructing Optimal Online Strategies

We now use the results above to construct an algorithm that produces the best possible online strategy for the multislope problem. The idea is to guess a competitive factor c, and then try to construct a c-competitive tight profile. Given a way to test for success, we can apply binary search to find the optimal competitive ratio c to any desired precision.

The main questions are how to test whether a given c is feasible, and how to construct the profiles. We answer these questions together: given c, we construct a tight c-competitive profile until either we fail (because c was too small) or until we can guarantee success. In the remainder of this section we describe how to construct a tight profile p for a given competitive factor c.

We begin with analyzing the way a tight profile may spend money. Consider the situation at some time t such that $p_k(t) < 1$. Let j be the maximum index such that $s_j \leq t$. Then $\frac{d}{dt}\text{OPT}(t) = r_j$. Therefore, the spending rate of a tight profile at time t must be $c \cdot r_j$. If $j < k$, the tight profile may spend at rate $c \cdot r_j$ until time s_{j+1} (or until $p_k(t) = 1$), and if

$j = k$ the tight profile may continue spending at this rate forever. Hence, for $t \in (s_j, s_{j+1})$, we have

$$\frac{d}{dt}B_p(t) + R_p(t) = c \cdot \frac{d}{dt}\text{OPT}(t) = c \cdot r_j \ . \tag{4.1}$$

Since p is tight and therefore prudent, we also have, assuming $\text{best}(t) = i$ and $\text{next}(t) = i+1$, that

$$B_p(t) = p_i(t)b_i + p_{i+1}(t)b_{i+1} \ ,$$

and

$$R_p(t) = p_i(t)r_i + p_{i+1}(t)r_{i+1} \ .$$

Plugging the above equations into Eq. (4.1), we get

$$\frac{d}{dt}p_i(t)b_i + \frac{d}{dt}p_{i+1}(t)b_{i+1} + p_i(t)r_i + p_{i+1}(t)r_{i+1} = c \cdot r_j$$

Since p is prudent, $p_i(t) = 1 - p_{i+1}(t)$ and hence $\frac{d}{dt}p_i(t) = -\frac{d}{dt}p_{i+1}(t)$. It follows that

$$\frac{d}{dt}p_{i+1}(t) + p_{i+1}(t) \cdot \frac{r_{i+1} - r_i}{b_{i+1} - b_i} = \frac{c \cdot r_j - r_i}{b_{i+1} - b_i} \tag{4.2}$$

A solution to a differential equation of the form $y'(x) + \alpha y(x) = \beta$ where α and β are constants is $y = \frac{\beta}{\alpha} + \Gamma \cdot e^{-\alpha x}$, where Γ depends on the boundary condition. Hence in our case we conclude that

$$p_{i+1}(t) = \frac{c \cdot r_j - r_i}{r_{i+1} - r_i} + \Gamma \cdot e^{\frac{r_i - r_{i+1}}{b_{i+1} - b_i} \cdot t} \ , \tag{4.3}$$

and $p_i(t) = 1 - p_{i+1}(t)$, where the constant Γ is determined by the boundary condition.

Eq. (4.3) is our tool to construct p in a piecewise iterative fashion. For example, we start constructing p from $t = 0$ using $p_1(t) = \frac{c \cdot r_0 - r_0}{r_1 - r_0} + \Gamma \cdot e^{\frac{r_0 - r_1}{b_1 - b_0} \cdot t}$ and the boundary condition $p_1(0) = 0$. We get that $\Gamma = \frac{r_0(c-1)}{r_0 - r_1}$, i.e.,

$$p_1(t) = \frac{r_0(c-1)}{r_0 - r_1} \cdot \left(e^{\frac{r_0 - r_1}{b_1 - b_0}t} - 1\right) \ ,$$

and this holds for all $t \leq \min(s_1, t_1)$, where t_1 is the solution to $p_1(t_1) = 1$.

In general, Eq. (4.2) remains true so long as there is no change in the spending rate and in the slope the profile p is buying. The spending rate changes when t crosses s_j, and the profile starts buying slope $i + 2$ when $p_{i+1}(t) = 1$.

We can now describe our algorithm. Given a ratio c, Algorithm FEASIBLE is able to construct the tight profile p or to determine that such a profile does not exist. It starts with the boundary condition $p_1(0) = 0$ and reveals the first part of the profile as shown above. Then, each time the spending rate changes or there is a change in $\text{best}(i)$ it moves to the next differential equation with a new boundary condition. After at most $2k$ such iterations it either computes a c-competitive tight profile p or discovers that such a profile is infeasible. Since we are able to test for success using Algorithm FEASIBLE, we can apply binary search to find the optimal competitive ratio to any desired precision.

We note that it is easy to construct a strategy that corresponds to any given prudent profile p, as described in the proof of Lemma 3.3. We conclude with the following theorem.

Theorem 4.6. *There exists an $O(k \log \frac{1}{\varepsilon})$ time algorithm that given an instance of the additive multislope ski rental problem for which the optimal randomized strategy has competitive ratio c, computes a $(c + \varepsilon)$-competitive strategy.*

Algorithm 1 – FEASIBLE$(c, \mathcal{M})$: true if the k-ski instance $\mathcal{M} = (b, r)$ admits competitive factor c

---

1: Let $s_i = \frac{b_i - b_{i-1}}{r_{i-1} - r_i}$ for each $1 \le i \le k$
2: Boundary_Condition $\leftarrow$ "$p_1(0) = 0$"
3: $j \leftarrow 0; i \leftarrow 1$
4: **loop**
5:     Define $p_i(t) = \frac{c \cdot r_j - r_{i-1}}{r_i - r_{i-1}} + \Gamma \cdot \exp(\frac{r_{i-1} - r_i}{b_i - b_{i-1}} \cdot t)$
6:     Try to solve for Γ using Boundary_Condition
7:     **if** no solution **then return** FALSE   ▷ *possible escape if not feasible*
8:     $y \leftarrow p_i(s_j)$
9:     **if** $y < 1$ **then**
10:       Boundary_Condition $\leftarrow$ "$p_i(s_j) = y$"
11:       $j \leftarrow j + 1$ ▷ *continue at the next interval* $[s_j, s_{j_1}]$
12:     **else**
13:       Let x be such that $p_i(x) = 1$
14:       Boundary_Condition $\leftarrow$ "$p_{i+1}(x) = 0$"
15:       $i \leftarrow i + 1$ ▷ *move to next slope*
16:     **end if**
17:     **if** $i > k$ or $j \ge k$ **then return** TRUE   ▷ *we're done*
18: **end loop**

---

5. An e-Competitive Strategy for the Non-Additive Case

In this section we consider the non-additive multislope ski rental problem. We present a simple randomized strategy which improves the best known competitive ratio from $2/\ln 2 = 2.88$ to e. Our technique is a simple application of randomized repeated doubling (see, e.g., [7]), used extensively in competitive analysis of online algorithms. For example, deterministic repeated doubling appears in [1], and a randomized version appears in [13].

Before presenting the strategy let us consider the details of the non-additive model. Augustine at el. [2] define a general non-additive model in which a transition cost b_{ij} is associated with every two states i and j, and show that one may assume w.l.o.g. that $b_{ij} = 0$ if $i > j$ and that $b_{ij} \le b_j$ for every $i < j$. Observe that we may further assume that $b_{ij} = b_j$ for every i and j, since the optimal (offline) strategy remains the unchanged. It follows that the strategies from [3, 4, 6] that were designed for the case of buying slopes "from scratch" also work for the general non-additive case.

We propose using the following iterative online strategy, which is similar to the one in [6], except for the choice of the "doubling factor." Specifically, the jth iteration is associated with a bound B_j on $\text{OPT}(\tau)$, where τ denotes the termination time of the game. We define $B_1 \stackrel{\text{def}}{=} \text{OPT}(s_1)/\alpha^X$, where $\alpha > 1$ and X is a chosen at random uniformly in $[0, 1)$. We also define $B_{j+1} = \alpha \cdot B_j$. Let $\tau_j = \text{OPT}^{-1}(B_j)$ and let i_j be the optimal offline state at time τ_j. In case there are two such states, i.e., $\tau_j = s_i$ for some i, we define $i_j = i - 1$. It follows that $i_1 = 0$. In the beginning of the jth iteration the online strategy buys i_j and stays in i_j until the this iteration ends. The jth iteration ends at time τ_j. Observe that the first iteration starts with $B_1 = \text{OPT}(s_1)$, namely we use slope 0 until s_1.

Theorem 5.1. *The expected cost of the strategy described above is at most e times the optimum.*

Proof. Observe that the first iteration starts with $B_1 = \mathrm{OPT}(s_1)$, namely we use slope 0 until s_1, and hence, if the game ends during the first iteration, i.e., before s_1/α^X, then the online strategy is optimal. Consider now the case where the game ends at time $\tau \geq s_1/\alpha^X$, and suppose that $\tau \in [\tau_\ell, \tau_{\ell+1})$ for $\ell > 1$. In this case, the expected cost of the online strategy is bounded by

$$
\mathbf{E}\left[\sum_{j=1}^{\ell} \mathrm{OPT}(\tau_j) + \mathrm{OPT}(\tau)\right] \leq \mathbf{E}\left[\sum_{j=1}^{\ell+1} \mathrm{OPT}(\tau_j)\right] \leq \mathbf{E}\left[\frac{\alpha}{\alpha - 1} \cdot \mathrm{OPT}(\tau_{\ell+1})\right]
$$

$$
= \mathbf{E}\left[\frac{\alpha^{2-X}}{\alpha - 1} \cdot \mathrm{OPT}(\tau)\right]
$$

$$
= \frac{\alpha}{\alpha - 1} \cdot \int_{x=0}^{1} \alpha^x dx \cdot \mathrm{OPT}(\tau) = \frac{\alpha}{\ln \alpha} \cdot \mathrm{OPT}(\tau)
$$

By choosing $\alpha = e$ the competitive ratio is $\frac{\alpha}{\ln \alpha} = e$ as required. ∎

Acknowledgment

References

[1] J. Aspnes, Y. Azar, A. Fiat, S. A. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM*, 44(3):486–504, 1997.

[2] J. Augustine, S. Irani, and C. Swamy. Optimal power-down strategies. In *45th IEEE Symp. on Foundations of Computer Science*, pages 530–539, 2004.

[3] Y. Azar, Y. Bartal, E. Feuerstein, A. Fiat, S. Leonardi, and A. Rosén. On capital investment. *Algorithmica*, 25(1):22–36, 1999.

[4] Y. Bejerano, I. Cidon, and J. S. Naor. Dynamic session management for static and mobile users: a competitive on-line algorithmic approach. In *4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 65–74. ACM, 2000.

[5] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[6] P. Damaschke. Nearly optimal strategies for special cases of on-line capital investment. *Theoretical Computer Science*, 302(1-3):35–44, 2003.

[7] S. Gal. *Search Games*. Academic Press, 1980.

[8] S. Irani, R. K. Gupta, and S. K. Shukla. Competitive analysis of dynamic power management strategies for systems with multiple power savings states. In *Design, Automation and Test in Europe Conference and Exhibition*, pages 117–123, 2002.

[9] A. R. Karlin, C. Kenyon, and D. Randall. Dynamic TCP acknowledgment and other stories about $e/(e-1)$. *Algorithmica*, 36(3):209–224, 2003.

[10] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):77–119, 1988.

[11] A. W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and Its Applications*. Academic Press, 1979.

[12] A. Meyerson. The parking permit problem. In *46th IEEE Symp. on Foundations of Computer Science*, pages 274–284, 2005.

[13] R. Motwani, S. Phillips, and E. Torng. Non-clairvoyant scheduling. *Theor. Comput. Sci.*, 130(1):17–47, 1994.