The Semi-stochastic Ski-rental Problem

Aleksander Mądry[1] and Debmalya Panigrahi[2]

1 **Microsoft Research New England**
Cambridge, MA, USA
madry@mit.edu

2 **Computer Science and Artificial Intelligence Laboratory**
Massachusetts Institute of Technology
Cambridge, MA, USA
debmalya@mit.edu

Abstract

In this paper, we introduce the *semi-stochastic* model for dealing with input uncertainty in optimization problems. This model is a hybrid between the overly pessimistic online model and the highly optimistic stochastic (or Bayesian) model. In this model, the algorithm can obtain only *limited* stochastic information about the future (i.e. about the input distribution)—as the amount of stochastic information we make available to the algorithm grows from no information to full information, we interpolate between the online and stochastic models. The central question in this framework is the trade-off between the performance of an algorithm, and the stochastic information that it can access. As a first step towards understanding this trade-off, we consider the *ski-rental* problem in the semi-stochastic setting. More precisely, given a desired competitive ratio, we give upper and lower bounds on the amount of stochastic information required by a deterministic algorithm for the ski-rental problem to achieve that competitive ratio.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases online optimization, stochastic algorithm

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2011.300

1 Introduction

In many real-world optimization problems, the input to the problem is not known at the outset, but is revealed slowly during the execution of the optimization algorithm. For example, in the *facility location* problem, where the goal is to select a minimum-cost set of locations for setting up facilities that can serve all clients, the exact set of clients is often not known in advance. Traditionally, such problems have been modeled by the *online* framework, where the algorithm is required to satisfy a (growing) set of constraints, while attempting to optimize the objective function. For example, in the facility location problem, the algorithm will need to ensure that at any stage, the facilities set up can serve the client requests received till that stage, while minimizing the cost of facilities. The performance of online algorithms is usually measured using the notion of *competitive ratio*, which is the maximum — taken over all the possible input sequences — of the ratio of the objective function in the solution produced by the algorithm to that of the optimal offline solution (i.e. the optimal solution when the entire input is known in advance).

After an initial period of vibrant research activity in online algorithms (see [1] for a survey), there has been a slight lull in recent years, partly due to the inherent pessimism of the online model, which rendered a lot of natural optimization tasks unnecessarily hard. For example, in the facility location problem, while the exact composition of the clientele is typically not

31st Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011).
Editors: Supratik Chakraborty, Amit Kumar; pp. 300–311

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

known in advance, a rough estimate of the clientele distribution is possible to obtain from e.g. market surveys. This has motivated the development of the *stochastic optimization* model (sometimes called the *Bayesian* model) in recent times, where the probability distribution over possible input sequences is known in advance, though the exact input sequence is unknown. The performance of such online algorithms is typically measured using one of the following two metrics:

- **Expectation of ratio (EoR).** The expected value of the competitive ratio of the algorithm when the input is drawn from the worst-case distribution.
- **Ratio of expectation (RoE).** The ratio of the expected value of the objective function in the algorithmic solution to the expected value of the objective function in the offline optimal solution, when the input is drawn from the worst-case distribution.

In recent years, algorithms have been developed for stochastic versions of various fundamental optimization problems (see related work for some examples). However, in many situations, the stochastic optimization setting is overly optimistic. For example, in the facility location problem described above, while it is possible to obtain a rough estimate of the clientele distribution, such information typically comes from market surveys which are expensive to conduct. In particular, to obtain fine-grained information, one needs to conduct surveys using a large sample, thereby incurring significant cost. In essence, there is a cost associated with obtaining stochastic information about the input for an online optimization problem, and this cost grows as the algorithm seeks more detailed information about the distribution from which the input is drawn. In this correspondence, we take a step toward understanding this trade-off between the performance of an online algorithm and the cost it needs to pay to obtain stochastic information about the input.

We propose a model for stochastic optimization of online problems based on the following two-stage game between the *algorithm* and the *adversary*. In the first stage, which comprises multiple rounds, the algorithm gathers stochastic information from the adversary about the distribution of the input. Each round comprises a *query* asked by the algorithm about the input distribution that the adversary answers. The algorithm is *adaptive*, i.e. it can choose its next query based on the answers given by the adversary to its previous queries. However, the maximum number of queries that the algorithm can ask is given in advance — we call this the *query budget* of the algorithm. The first stage ends when the algorithm has exhausted its query budget. The second stage comprises an instance of the online problem where the adversary now presents the actual input sequence, which must be drawn from some distribution that is consistent with the stochastic information provided during the first stage. This allows the algorithm to use the information it acquired in the first stage to guide its choices. We call this the *semi-stochastic* model for optimization problems.

As in stochastic optimization, the performance of an algorithm in the semi-stochastic model can be measured using either the EoR or the RoE metrics. However, we need to be careful about how we define a *worst-case* distribution. The performance of the algorithm for a fixed distribution over the input space is defined as its competitive ratio for the worst-case sequence of (truthful) answers given by the adversary to the queries asked by the algorithm in the first stage. In other words, given a fixed distribution over the inputs that the adversary knows but the algorithm does not, one can visualize the first stage as a game where the algorithm aims to minimize its expected competitive ratio (which is measured using either the EoR or the RoE metric) and the adversary aims to maximize it. The overall performance of the algorithm is the worst-case performance over all distributions over the input space, where the performance for a fixed distribution is as described above.

The semi-stochastic model may be viewed as a hybrid between the pessimistic online

model and the optimistic stochastic model. In particular, if the algorithm has zero query budget, then we obtain the online model, while if the query budget is infinite, then we get the stochastic model. The goal of introducing this model is to study the trade-off between the query budget given to an online algorithm and its performance. As described earlier, a large query budget typically implies a large cost of obtaining such fine-grained information about the input distribution and might not be useful unless it improves the performance of the algorithm significantly.

It is instructive at this point to compare the semi-stochastic model to the well-known computational learning paradigm. The goal in computational learning is to learn a *hypothesis class* (i.e. a subspace of some fixed space) given the labeling of a few sample points from this space (i.e. whether each point is in the subspace or not). Typically, one is interested in the trade-off between the size of the sample and the accuracy of the guessed hypothesis. At a high level, one may interpret the semi-stochastic model as an extension of the computational learning paradigm to optimization problems. Here, our goal is to *learn* the input distribution to an optimization problem for the purpose of designing an algorithm with small competitive ratio while minimizing the amount of information about the distribution that we explicitly ask for in the form of queries.

It is important to note that we have not specified the exact nature of queries that the adversary is allowed to ask in the first stage. In principle, a particular instantiation of our model might allow for extremely powerful queries thereby letting a semi-stochastic algorithm gain more information about the input distribution than a corresponding stochastic algorithm that has access to the input distribution via a computationally bounded process, e.g. a polynomial number of samples. However, we expect that in most interesting instantiations of the semi-stochastic model, including the one that we will consider in this paper, the algorithm will be limited to some class of *computationally bounded queries*, i.e. queries that can be answered (to the desired accuracy) using polynomial amount of information regarding the input distribution.

1.1 An application: The ski-rental problem

Perhaps the most fundamental of online problems, the *ski-rental* problem is defined as follows. Suppose one wants to go skiing, but does not know the exact number of days that she would like to ski. Given this uncertainty, she needs to decide every morning whether she would like to buy a pair of skis that cost b for the entire season, or rent skis for the day paying 1. Overall, she would like to minimize her expenditure. The ski-rental problem (or its slight variants) have been used to solve various online problems, e.g. TCP acknowledgement, Bahncard problem, etc [14]. It is well-known that the ski-rental problem has a simple 2-competitive deterministic algorithm, and this is optimal for deterministic algorithms. Karlin *et al* [15] showed that the competitive ratio can be improved to $\frac{e}{e-1}$ in expectation, if the online algorithm is allowed to be randomized, and that this is optimal for randomized algorithms.

In this paper, we analyze the ski-rental problem in the semi-stochastic model as a first step towards investigating more complicated online problems. First, we note that if we employ the RoE performance metric in this problem, then we encounter the following difficulty. The adversary can create a distribution where an arbitrarily small probability mass is at $x \neq 0$ and all the remaining probability mass is at 0. (Unless otherwise stated, x will represent the duration of skiing which is the only input parameter for the ski-rental problem.) Then, in any reasonable query model, the algorithm will require an unbounded query budget to achieve any RoE value less than the trivial 2. We therefore focus on the EoR metric; in the remainder of the paper, the competitive ratio of an algorithm in the semi-stochastic model

will refer to the EoR metric. Further, as is standard in the literature (see e.g. [14]), we consider a continuous version of this problem. This allows us, in particular, to scale the price of the skis and assume wlog that $b = 1$—we adopt this convention from now on. The input is represented by a probability density function $p(x)$ such that $\int_0^\infty p(x)d(x) = 1$. Any deterministic strategy for this problem is parameterized by the value $k \geq 0$ such that skis are rented as long as $x < k$ and bought at $x = k$. We denote this strategy by $S(k)$, and its competitive ratio is given by

$$r(k) := \begin{cases} \int_0^k p(x)dx + (1+k)\int_k^\infty \frac{p(x)}{\min\{x,1\}}dx & \text{if } k \leq 1 \\ \int_0^1 p(x)dx + \int_1^k xp(x)dx + (1+k)\int_k^\infty p(x)dx & \text{if } k > 1. \end{cases}$$

Before proceeding further, we need to define our query model, i.e. the allowed set of queries that the algorithm can ask the adversary in the first stage. We consider the setting where each query is a quantile query, i.e. the algorithm gives a value $0 < y < 1$, and the adversary in response gives a number ℓ such that $\int_0^\ell p(x)dx = y$. Note that after the algorithm has asked a set of q queries, it can partition $[0, \infty)$ into a set of $q+1$ non-overlapping intervals such that the probability mass in each interval is known but the distribution within an interval is unknown to the algorithm. If the algorithm asks a large number of queries, then these intervals are fine-grained and therefore the algorithm has detailed information about the input distribution whereas if its query budget is small, then it obtains a very coarse-grained description of this distribution.

Note that the existence of the $\frac{e}{e-1}$-competitive randomized algorithm implies that for any input distribution, there exists a deterministic strategy $S(k)$ that achieves a competitive ratio of $\frac{e}{e-1}$. If not, then the corresponding distribution can be used with Yao's minimax principle [23] to prove a randomized lower bound greater than $\frac{e}{e-1}$ contradicting the algorithm of Karlin *et al.* Our goal, in this paper, is to determine the minimum query budget of a deterministic semi-stochastic algorithm that has a competitive ratio of $\frac{e}{e-1} + \epsilon$ for any $\epsilon > 0$. (Of course, we are interested in small fractions ϵ, since there is an online deterministic strategy that has a competitive ratio of 2.) To this end, we prove the two following theorems.

▶ **Theorem 1** (Lower Bound). *There exists a universal constant $C > 0$ such that for any constant $\varepsilon \in \left(0, \frac{e-2}{4(e-1)}\right]$,[1] if the algorithm has a quota of $q < \frac{C}{\varepsilon}$ queries in the first stage, then for any deterministic strategy $S(k)$ chosen in the second stage, there exists an input distribution $\mathcal{D}$ that has the following properties:*

- *The distribution $\mathcal{D}$ has the same quantiles as those provided by the adversary in the first stage.*
- *The expected competitive ratio of the algorithm $S(k)$ for input distribution $\mathcal{D}$ is greater than $\frac{e}{e-1} + \varepsilon$.*

▶ **Theorem 2** (Upper Bound). *For any $\epsilon > 0$, there is a polynomial-time deterministic algorithm with query budget $O\left(\frac{1}{\epsilon^{3/2}}\right)$ that has a competitive ratio of $\frac{e}{e-1} + \epsilon$.*

We find the lower bound somewhat surprising. Note that for most input distributions, an algorithm needs to know only some parts of the distribution in detail, but can do with more coarse-grained information about the other parts. Therefore, it is conceivable that an

[1] The range of ε in the theorem can be increased from $\left[0, \frac{e-2}{4(e-1)}\right]$ to $\left[0, \frac{\alpha(e-2)}{e-1}\right]$ for any $\alpha < 1$ with the only change being in the value of the constant C. At $\alpha = 1$, the lower bound must fail since there exists a 2-competitive deterministic online algorithm. Thus, $\lim_{\alpha \to 0} C = \infty$. Since we are primarily interested in the asymptotic behavior of the lower bound for small ε, and to maintain simplicity of notation, we consider the representative value of $\alpha = 1/4$ in the rest of the paper.

adaptive algorithm recognizes which parts of the distribution it needs to query in greater detail and therefore makes do with less than polynomial in $1/\epsilon$ queries overall. However, our lower bound proves that even when the algorithm is adaptive, it still requires a large (polynomial in $1/\varepsilon$) number of queries to pinpoint — up to an accuracy of ε — the parts of the input distribution that are more critical.

1.2 Our Techniques

Our algorithm for proving the upper bound obtains a set of equi-spaced quantiles in the first stage by asking the corresponding queries. Once it has obtained these quantiles, it assumes an arbitrary density function on each of the inter-quantile intervals, and computes the optimal deterministic algorithm for this assumed input distribution. We would then need to claim that no matter how much the actual density function in each inter-quantile interval differs from the assumed one, the performance of the algorithm does not degrade significantly. It turns out that for most natural choices of the assumed density function in each interval, this claim does not hold, i.e. the adversary can change the input distribution while not changing any of the quantiles to degrade the competitive ratio of the algorithm significantly. However, our crucial observation is that the claim would hold if the value of k in the deterministic strategy is large enough, i.e. the adversary can significantly degrade the performance of the algorithm only if it selects a small value of k. To make the algorithm robust to this possibility, our algorithm deliberately chooses a large enough value of k that might be sub-optimal for the assumed distribution. To prove that the sub-optimality arising from this choice of k can be absorbed in our competitive ratio, we show for any input distribution, there always exists a large enough value of k that is *almost* optimal.

Interestingly, our proof of the lower bound is somewhat related to the problem we highlighted above. Namely, it is established by making the adversary answer all the queries of the algorithm according to a certain distribution that penalizes strategies that buy skis later, forcing the algorithm to choose a strategy that buys relatively early. Next, we show that in this case, unless the algorithm is able to pinpoint the distribution beyond k (where the algorithm's chosen strategy is $S(k)$) up to accuracy $O(\varepsilon)$, the adversary can always modify the distribution to make the algorithm suffer a large competitive ratio, while retaining compatibility with the stochastic information provided earlier to the algorithm.

1.3 Related Work

The term *stochastic optimization* has been used to mean various algorithmic models in the literature. As described above, we use it to imply that the algorithm knows the distribution of the input but not the input itself. Various optimization problems have been studied in this model previously, including facility location [19], network design problems [20, 9], secretary-type problems [2, 17, 16, 18], matching [10, 8, 5, 3], packing and covering problems [6, 7, 11], etc. Another closely related model where significant progress has been reported in recent years is two-stage stochastic optimization with recourse (see [21] for a survey). Here, the algorithm has a choice of buying resources at a lower cost with only stochastic knowledge of the input, or at a higher cost with knowledge of the actual input. Various other models that lie between the online model and the stochastic model have also been considered in the literature, especially for online problems with multiple inputs. Among these, the two most important models are perhaps the i.i.d. from unknown distribution model where each input is drawn independently and identically from a distribution that is not known to the algorithm, and the random permutation model where the input sequence is a random permutation on

an adversarially chosen input set. In the ski-rental problem, there is only one input, and therefore, these multi-input models are not relevant.

A class of problems that have some similarity with the questions that we are exploring are multi-armed bandit (MAB) problems (see e.g. [4]). These problems model the classical "exploration-vs-exploitation" trade-off in decision problems. The particular variant of this problem that is most similar to our problem is the *budgeted learning* problem [12, 13], where a budgeted exploration phase is followed by an exploitation phase. However, the typical setup in these problems is that one has to choose from multiple options, each of which has an unknown associated reward, and the algorithm needs to explore the options within a stipulated budget so that it can maximize revenue (or minimize regret) later in the exploitation phase. On the other hand, our model pertains to optimization problems, and the goal of the exploration phase is to gain sufficient knowledge about the input space for optimizing the objective function in the exploitation phase.

1.4 Roadmap

In section 2, we prove the lower bound on the query complexity of the ski-rental problem in the semi-stochastic model (Theorem 1); the corresponding upper bound (Theorem 2) appears in section 3. Finally, in section 4, we give some directions for future work on the semi-stochastic model.

2 A Lower Bound on the Competitive Ratio of Semi-Stochastic Algorithms for the Ski-rental Problem

In this section, our goal is to prove Theorem 1, i.e. to give a lower bound on the number of queries that *any* semi-stochastic algorithm needs to ask in order to have an expected competitive ratio of at most $\frac{e}{e-1}+\epsilon$. As earlier, we will assume wlog that the cost of buying skis is 1, and $S(k)$ will represent the deterministic strategy that rents skis until the duration of skiing x reaches k, at which point skis are bought.

Consider a probability distribution $\mathcal{D}_0$ on the input given by the following density function for $0 \leq x \leq 1$: $\mathcal{D}_0(x) = \left(\frac{e}{e-1}\right) xe^{-x}$. In addition, we have a probability mass of $\frac{1}{e-1}$ at $x = +\infty$.[2] For any $0 \leq \delta < 1$, consider the distribution $\mathcal{D}_\delta$ created by scaling the density function of $\mathcal{D}_0$ by $1-\delta$ in the range $0 \leq x \leq 1$, and shifting the surplus probability mass of $\delta\left(\frac{e-2}{e-1}\right)$ to $+\infty$. The density function for this distribution in the range $0 \leq x \leq 1$ is given by $\mathcal{D}_\delta(x) = (1-\delta)\left(\frac{e}{e-1}\right) xe^{-x}$. In addition, there is a probability mass of $\frac{1+(e-2)\delta}{e-1}$ at $x = +\infty$. The next lemma lower bounds the expected competitive ratio of deterministic strategies for distribution $\mathcal{D}_\delta$.

▶ **Lemma 3.** *For any $k \geq 0$ and $0 \leq \delta < 1$, the expected competitive ratio of strategy $S(k)$ where the input has distribution $\mathcal{D}_\delta$ is $\frac{e}{e-1} + \delta\left(k - \frac{1}{e-1}\right)$ if $0 \leq k \leq 1$ and at least $\frac{e}{e-1} + \delta k(1 - \frac{1}{e-1})$ if $k > 1$.*

[2] We can use Dirac delta function [22] to define the density function at $x = +\infty$ in a mathematically precise manner, but we will ignore this technicality throughout the paper and assume a probability mass of $\frac{1}{e-1}$ at $x = +\infty$.

Proof. For $0 \leq k \leq 1$, the expected competitive ratio of strategy $S(k)$ is

$$\frac{e(1-\delta)}{e-1}\left(\int_0^k xe^{-x}dx + \int_k^1 (1+k)e^{-x}dx\right) + \left(\frac{1+k}{e-1}\right)(1+\delta(e-2)) = \frac{e}{e-1} + \delta\left(k - \frac{1}{e-1}\right).$$

On the other hand, if $k > 1$, then the expected competitive ratio of strategy $S(k)$ is

$$\int_0^1 \frac{e(1-\delta)}{e-1}xe^{-x}dx + \left(\frac{1+k}{e-1}\right)(1+\delta(e-2)) \geq \frac{e}{e-1} + \delta k\left(1 - \frac{1}{e-1}\right). \quad \blacktriangleleft$$

This lemma implies that as δ grows, the distribution $\mathcal{D}_\delta$ favors strategies $S(k)$ for small values of k by making their expected competitive ratio less than $\frac{e}{e-1}$, while making the expected competitive ratio of strategies $S(k)$ with large values of k worse. This property of distribution $\mathcal{D}_\delta$ will be crucial in our lower bound construction.

We are now ready to prove our lower bound. We will use the following fact.

▶ Fact 1. For any $t \geq 0$ and $0 \leq \Delta \leq 1$, $\int_t^{t+\Delta}\left(\frac{x}{t} - 1\right)e^{-x}dx \geq \frac{e^{-t}}{t}\Delta^2$.

Proof. We have

$$\int_t^{t+\Delta}\left(\frac{x}{t} - 1\right)e^{-x}dx = \frac{e^{-t}}{t}\left(1 - e^{-\Delta}(1+\Delta)\right) \geq \frac{e^{-t}}{t}\left(1 - (1-\Delta)(1+\Delta)\right) = \frac{e^{-t}}{t}\Delta^2. \quad \blacktriangleleft$$

Proof of Theorem 1. In the first round, for any query asked by the algorithm, the adversary returns the corresponding quantile of the distribution $\mathcal{D}_\delta$ for some δ whose value (that will depend only on ε) we will fix later. This partitions the entire input interval $[0, \infty)$ into contiguous non-overlapping intervals with the quantiles representing the boundaries between adjacent intervals. Let these quantiles be $y_0 = 0, y_1, \ldots, y_q$, where $y_i \leq y_{i+1}$ for each i. Note that for each interval $[y_i, y_{i+1}]$, the algorithm knows the probability that the input is contained in it, though it does not know the exact distribution.

For notational convenience, we assume wlog that the strategy $S(k)$ chosen by the algorithm satisfies $k = y_s$ for some $0 \leq s \leq q$.[3] Based on the values of y_is and the choice of k, the adversary chooses the actual distribution $\mathcal{D}$ of the input. This distribution is obtained from $\mathcal{D}_\delta$ by concentrating the entire probability mass in each interval (y_i, y_{i+1}) where $i \geq s$, at a value y_i^+ that is infinitesimally close to (but greater than) y_i, while leaving the probability distribution of $x < y_s$ and that of $x = +\infty$ unchanged. Formally, the input distribution $\mathcal{D}$ is given by the following density function, where y_{q+1} is any finite value that is greater than $\max\{1, y_q\}$:

$$\mathcal{D}(x) = \begin{cases} \int_{y_i^+}^{y_{i+1}} \mathcal{D}_\delta(x)dx & \text{for each } x = y_i^+, \text{ where } s \leq i \leq q \\ \mathcal{D}_\delta(x) & \text{if } x \leq y_s \text{ or } x = +\infty \\ 0 & \text{otherwise.} \end{cases}$$

Note that by construction, the quantiles $y_1, y_2, \ldots, y_q$ hold for distribution $\mathcal{D}$ as well.

First, note that the expected competitive ratio of strategy $S(y_s)$ for distribution $\mathcal{D}_\delta$ is at most as much as that for distribution $\mathcal{D}$. Further, the difference between these expected

[3] If the algorithm chooses $y_s < k < y_{s+1}$ for some s, then the adversary uses the same distribution as that for $k = y_s$ except that the probability mass in the interval (y_s, y_{s+1}) is concentrated at x^+ rather than at y_s^+. Clearly, the expected competitive ratio of the algorithm for this input distribution is worse than the expected competitive ratio for the corresponding input distribution if the algorithm had chosen $k = y_s$.

competitive ratios is only due to the difference in the probability density function in the range $y_s < x \leq 1$. Let y_r be the maximum value of y_i that is less than 1. The difference in the expected competitive ratios is then given by

$$\begin{aligned} & (1-\delta)\left(\frac{e}{e-1}\right)\left(\sum_{i=s}^{r-1}(1+y_s)\int_{y_i}^{y_{i+1}}\left(\frac{1}{y_i}-\frac{1}{x}\right)xe^{-x}dx + \int_{y_r}^{1}\left(\frac{1}{y_i}-\frac{1}{x}\right)xe^{-x}dx\right) \\ = \quad & \frac{e(1-\delta)}{e-1}\sum_{i=s}^{q}\int_{z_i}^{z_{i+1}}\left(\frac{x}{z_i}-1\right)e^{-x}dx, \end{aligned} \tag{1}$$

where $z_i = \min(y_i, 1)$.

We now have two cases. First, suppose $y_s > \frac{e}{2(e-1)}$. Then, the expected competitive ratio of strategy $S(y_s)$ when the input has distribution $\mathcal{D}$ is at least the expected competitive ratio when the input has distribution $\mathcal{D}_\delta$, which in turn is greater than $\frac{e}{e-1} + \frac{e-2}{2(e-1)}\delta$ by Lemma 3. Now, let $\delta = \frac{2(e-1)}{e-2}\varepsilon$; then the ratio is greater than $\frac{e}{e-1} + \varepsilon$.

The other case is when $y_s \leq \frac{e}{2(e-1)}$. Then, equation (1) and Lemma 3 imply that the expected competitive ratio of strategy $S(y_s)$ when the input has distribution $\mathcal{D}$ is at least

$$\begin{aligned} & \frac{e}{e-1} + \delta\left(y_s - \frac{1}{e-1}\right) + \frac{e(1-\delta)}{e-1}\sum_{i=s}^{q}\int_{z_i}^{z_{i+1}}\left(\frac{x}{z_i}-1\right)e^{-x}dx \\ \geq \quad & \frac{e}{e-1} - \frac{\delta}{e-1} + \frac{e(1-\delta)}{e-1}\sum_{i=s}^{q}\frac{e^{-z_i}}{z_i}\Delta_i^2, \quad \text{where } \Delta_i := z_{i+1} - z_i \geq 0 \\ \geq \quad & \frac{e}{e-1} - \frac{2\varepsilon}{e-2} + \frac{1}{2(e-1)}\sum_{i=s}^{q}\Delta_i^2 \\ \geq \quad & \frac{e}{e-1} - \frac{2\varepsilon}{e-2} + \frac{1}{2(e-1)}\frac{\left(\sum_{i=s}^{q}\Delta_i\right)^2}{q-s} \\ \geq \quad & \frac{e}{e-1} - \frac{2\varepsilon}{e-2} + \frac{(e-2)^2\varepsilon}{8(e-1)^3 C}. \end{aligned}$$

The first inequality follows from Fact 1; the second one from $z_i \leq 1$ and $\delta = \frac{2(e-1)}{e-2}\varepsilon$; the third one from $1 - \delta \geq 1/2$ since $\varepsilon \leq \frac{e-2}{4(e-1)}$; the fourth one from the fact that for any set of l numbers $d_1, \ldots, d_l$, $\sum_{i=1}^{l} d_i^2 \geq \frac{\left(\sum_{i=1}^{l} d_i\right)^2}{l}$; and the fifth one from $\sum_{i=s}^{q}\Delta_i \geq \frac{e-2}{2(e-1)}$ since $y_s \leq \frac{e}{2(e-1)}$, and from $q - s \leq q \leq C/\varepsilon$. Finally, we choose $C < \frac{1}{8e}\left(\frac{e-2}{e-1}\right)^3$ to ensure that the above competitive ratio is greater than $\frac{e}{e-1} + \varepsilon$. ◀

3 A Semi-stochastic Algorithm for the Ski-rental Problem

In this section, we will prove Theorem 2 by giving an algorithm that asks $O\left(\frac{1}{\epsilon^{3/2}}\right)$ quantile queries in the first stage, and then chooses a deterministic strategy (based on the quantiles revealed in the first stage) that has an expected competitive ratio of $\frac{e}{e-1} + \epsilon$. We describe the algorithm with a parameter δ that we will fix later. As mentioned in the introduction, we will assume wlog that the cost of buying skis is 1. Recall that any deterministic strategy can be described by a single parameter k which represents the strategy of renting skis until the duration of skiing x reaches k. At that point, skis are bought. As earlier, we denote this strategy by $S(k)$.

- In the first stage, the algorithm asks $1/\delta - 1$ queries for quantiles $\delta, 2\delta, \ldots, 1-\delta$. This partitions the input space $[0, \infty)$ into a set of $1/\delta$ contiguous, non-overlapping intervals, each of which has a probability mass of δ.
- The algorithm now assumes that the probability mass of each interval described above is concentrated at the right boundary of the interval, i.e. at its maximum value. For the last interval, this corresponds to assuming that there is δ mass of the input distribution at infinity.
- With the above assumption about the distribution, the algorithm outputs the deterministic strategy $S(k)$ that minimizes the expected competitive ratio subject to the constraint that $\delta^{1/3} \leq k \leq 1$. The only possible values of k are the values of the quantiles that are in the above range and $\delta^{1/3}$. Therefore, k can be found in time polynomial in $1/\delta$.

We will now analyze the expected competitive ratio of this strategy $S(k)$. We will use the following fact.

▶ Fact 2. For any $0 \leq a \leq 1$, $1 + \frac{1}{(1+a)e^{1-a}-1} \leq \frac{e}{e-1}(1+a^2)$.

Proof. We have

$$1+\frac{1}{(1+a)e^{1-a}-1} = \frac{e(1-(1+a)e^{-\alpha})}{((1+a)e^{1-a}-1)(e-1)}+\frac{e}{e-1} \leq \frac{e}{e-1}\left(2-(1+a)e^{-a}\right) \leq \frac{e}{e-1}(1+a^2).$$

The penultimate step follows from that fact that $((1+a)e^{1-a}-1) \geq 1$ for $0 \leq a \leq 1$, while the last step follows from the fact that for any z, $e^{-z} \geq 1 - z$. ◀

The next theorem is crucial.

▶ **Theorem 4.** *For* any *distribution over the input, the best deterministic strategy $S(k)$ subject to the constraint that $\alpha \leq k \leq 1$ for some fixed $0 \leq \alpha \leq 1$ has an expected competitive ratio of at most $\frac{e}{e-1}(1+\alpha^2)$.*

Proof. Let the worst-case input distribution for an algorithm that selects the best strategy in the above range (i.e. the input distribution for which the competitive ratio of the algorithm is the worst) be called the *nemesis* distribution. Our goal is to construct a nemesis distribution and show that the expected competitive ratio of the algorithm is at most $\frac{e}{e-1}(1+\alpha^2)$ for this distribution.

Let $\mathcal{P} : [0, \infty) \to [0, 1]$ be the probability density function corresponding to the nemesis distribution. Our first claim is that the nemesis distribution has no probability mass in the range $[0, \alpha]$, i.e. $\int_0^\alpha \mathcal{P}(x)dx = 0$. Suppose not; then, let $\int_0^\alpha \mathcal{P}(x)dx = \delta > 0$. Let the expected competitive ratio of the algorithm for this input be r. Since the strategy $S(\gamma)$ for any $\gamma \geq \alpha$ is optimal for the probability mass of δ in the range $[0, \alpha]$, it follows that

$$\delta + \int_\alpha^\gamma \mathcal{P}(x)dx + (1+\gamma)\int_\gamma^1 \frac{\mathcal{P}(x)}{x}dx + (1+\gamma)\int_1^\infty \mathcal{P}(x)dx \geq r$$

for all $\alpha \leq \gamma \leq 1$. Now, consider an alternative distribution which is identical to the previous distribution, except that this probability mass of δ is shifted from the range $[0, \alpha]$ to the range $[1, \infty]$ (the exact change in the density function can be arbitrary as long as the above property is satisfied). Now, for any strategy $S(\gamma)$ satisfying $\alpha \leq \gamma \leq 1$ that the algorithm can produce, the competitive ratio is

$$\begin{aligned} & \int_\alpha^\gamma \mathcal{P}(x)dx + (1+\gamma)\int_\gamma^1 \frac{\mathcal{P}(x)}{x}dx + (1+\gamma)\int_1^\infty \mathcal{P}(x)dx + (1+\gamma)\delta \\ > \quad & \int_\alpha^\gamma \mathcal{P}(x)dx + (1+\gamma)\int_\gamma^1 \frac{\mathcal{P}(x)}{x}dx + (1+\gamma)\int_1^\infty \mathcal{P}(x)dx + \delta \\ \geq \quad & r. \end{aligned}$$

This contradicts the definition of a nemesis distribution; hence $\int_0^\alpha \mathcal{P}(x)dx = 0$.

Let $\mathcal{P}_\infty = \int_1^\infty \mathcal{P}(x)dx$. Then the expected competitive ratio of strategy $S(\gamma)$ (where $\alpha \leq \gamma \leq 1$) can be thought of as a function of γ,

$$r(\gamma) = \int_\alpha^\gamma \mathcal{P}(x)dx + (1+\gamma)\int_\gamma^1 \frac{\mathcal{P}(x)}{x}dx + (1+\gamma)\mathcal{P}_\infty.$$

It follows from standard arguments (see e.g. [15, 14]) that for a nemesis distribution, $r(\gamma)$ must be invariant of γ. Let $r'(\gamma) = \frac{dr}{d\gamma}$ and $r''(\gamma) = \frac{d^2r}{d\gamma^2}$. Then,

$$\begin{aligned} r'(\gamma) &= \int_\gamma^1 \frac{\mathcal{P}(x)}{x}dx - \frac{\mathcal{P}(\gamma)}{\gamma} + \mathcal{P}_\infty, \quad \text{and} \\ r''(\gamma) &= -\frac{\mathcal{P}(\gamma)}{\gamma} - \frac{\mathcal{P}'(\gamma)}{\gamma} + \frac{\mathcal{P}(\gamma)}{\gamma^2}, \end{aligned}$$

where $\mathcal{P}'(\gamma) = \frac{d\mathcal{P}}{d\gamma}$. For the invariance property to hold, $r''(\gamma) = r'(\gamma) = 0$, which gives $\mathcal{P}(\gamma) = \mathcal{P}_\infty \gamma e^{1-\gamma}$. Then, the overall probability mass is given by

$$\int_0^\infty \mathcal{P}(x)dx = e\mathcal{P}_\infty \int_\alpha^1 xe^{-x}dx + \mathcal{P}_\infty = -e\mathcal{P}_\infty\left(\frac{2}{e} - (1+\alpha)e^{-\alpha}\right) + \mathcal{P}_\infty = \mathcal{P}_\infty\left((1+\alpha)e^{1-\alpha} - 1\right).$$

Since $\int_0^\infty \mathcal{P}(x)dx = 1$, it follows that $\mathcal{P}_\infty = \frac{1}{(1+\alpha)e^{1-\alpha}-1}$. Therefore, the probability density function for the nemesis distribution is

$$\mathcal{P}(x) = \begin{cases} 0 & \text{if } x < \alpha \\ \left(\frac{1}{(1+\alpha)e^{1-\alpha}-1}\right) xe^{1-x} & \text{if } \alpha \leq x \leq 1. \end{cases}$$

There is also a probability mass of $\frac{1}{(1+\alpha)e^{1-\alpha}-1}$ in the range $[1, \infty]$; the exact density function in this range is inconsequential.

Note that by our derivation, the expected competitive ratio of any strategy in the range $[\alpha, 1]$ is identical; hence, the expected competitive ratio of the strategy $S(k)$ chosen by the algorithm is given by (we calculate the competitive ratio of $S(1)$)

$$1 - \mathcal{P}_\infty + 2\mathcal{P}_\infty = 1 + \mathcal{P}_\infty = 1 + \frac{1}{(1+\alpha)e^{1-\alpha}-1} \leq \frac{e}{e-1}(1+\alpha^2),$$

by Fact 2. ◀

We now use this theorem to obtain a bound on the competitive ratio of the algorithm given above.

▶ **Theorem 5.** *The competitive ratio of the algorithm given above is at most* $\frac{e}{e-1} + \frac{2e-1}{e-1}\delta^{2/3}$.

Proof. Consider a game between the algorithm and an adversary where the adversary needs to initially present the quantiles queried by the algorithm, then the algorithm outputs a strategy $S(k)$ according to the description above, and finally the adversary reveals the actual distribution that is consistent with the quantiles it displayed initially. The goal of the adversary is to maximize the expected competitive ratio of the algorithm for the distribution that she reveals at the end. In the last step, the adversary has freedom to define any probability density function in the interval between every pair of adjacent quantiles as long as the total probability mass in any such interval is δ. We consider the options available to the adversary in the three ranges $[0, k]$, $[k, 1]$ and $[1, \infty)$. The actual probability density function in the first and last of these ranges does not change the expected competitive ratio

of the strategy $S(k)$. However, in the range $[k, 1]$, the adversary would move the entire probability mass in each individual interval to the quantile at its left boundary, thereby maximally increasing the expected competitive ratio of $S(k)$. This would decrease the cost of the optimal offline solution to the maximum extent while not changing the cost of the algorithmic solution. The net effect of these changes is that the actual input distribution has a δ probability mass at k instead of at 1. Thus, the difference of the expected competitive ratio of strategy $S(k)$ for the actual input distribution and that for the assumed distribution is at most $\delta\left(\frac{1+k}{k} - (1+k)\right) \leq \frac{\delta}{k} \leq \delta^{2/3}$ since $k \geq \delta^{1/3}$. By the above theorem, the expected competitive ratio of strategy $S(k)$ for the assumed distribution is at most $\frac{e}{e-1}(1+\delta^{2/3})$. Therefore, the competitive ratio of the algorithm for the actual input distribution is at most $\frac{e}{e-1} + \frac{2e-1}{e-1}\delta^{2/3}$. ◀

Finally, we note that Theorem 2 follows from the above theorem by replacing δ with $\left[\left(\frac{e-1}{2e-1}\right)\epsilon\right]^{3/2}$.

4 Conclusion and Future Work

In this paper, we introduced a new model for online optimization problems that we call the semi-stochastic model. This framework offers a hybrid between the excessively pessimistic online model and the overly optimistic stochastic model. As a first step towards understanding optimization problems in this setting, we presented an algorithm for the ski-rental problem that gives a trade-off between the amount of stochastic information available to the algorithm and its performance. Perhaps more surprisingly, we also gave a lower bound on the query complexity for algorithms in this model that loosely matches the upper bound. Our lower bound holds for the powerful class of adaptive algorithms.

This initial result opens up the possibility of studying a plethora of online problems in the semi-stochastic model. Our model does not impose any restriction on the type of queries that the algorithm is allowed to ask of the adversary. So, it would be interesting to consider other query models as well. For example, as discussed in the introduction, in certain (e.g. computational learning) situations, natural queries might correspond to sampling inputs.

A somewhat different direction of future research would be to investigate the robustness of stochastic algorithms. One interpretation of our semi-stochastic algorithms is that they are robust to changes in the input distribution modulo the satisfaction of a set of parameters (corresponding to the answers given to the queries). Following up in this direction, it would be interesting to investigate the sensitivity of known stochastic algorithms to changes in the input distribution, and if they turn out to be sensitive, to design algorithms that are robust to such changes. The ultimate goal in this direction of research would be to obtain algorithms that degrade gracefully with changes in the input distribution, with the best stochastic algorithm and the best online algorithm being the two ends of the spectrum.

5 Acknowledgement

We would like to thank Sudipto Guha, Adam Kalai, Kamesh Munagala and R. Ravi for useful discussions on our problem. We also thank an anonymous referee for detailed comments that helped improve the presentation of the paper.

References

1 Susanne Albers and Stefano Leonardi. On-line algorithms. *ACM Comput. Surv.*, 31(3es):4, 1999.
2 Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA*, pages 434–443, 2007.
3 Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings - (extended abstract). In *ESA (2)*, pages 218–229, 2010.
4 Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.
5 Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *ICALP (1)*, pages 266–278, 2009.
6 Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Adaptivity and approximation for stochastic packing problems. In *SODA*, pages 395–404, 2005.
7 Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.*, 33(4):945–964, 2008.
8 Jon Feldman, Aranyak Mehta, Vahab S. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *FOCS*, pages 117–126, 2009.
9 Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *SODA*, pages 942–951, 2008.
10 Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, pages 982–991, 2008.
11 Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. Set covering with our eyes closed. In *FOCS*, pages 347–356, 2008.
12 Sudipto Guha and Kamesh Munagala. Approximation algorithms for budgeted learning problems. In *STOC*, pages 104–113, 2007.
13 Sudipto Guha and Kamesh Munagala. Model-driven optimization using adaptive probes. In *SODA*, pages 308–317, 2007.
14 Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic TCP acknowledgment and other stories about e/(e-1). *Algorithmica*, 36(3):209–224, 2003.
15 Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel Dominic Sleator. Competitive snoopy caching. *Algorithmica*, 3:77–119, 1988.
16 Robert Kleinberg. Geographic routing using hyperbolic space. In *INFOCOM*, pages 1902–1909, 2007.
17 Robert D. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA*, pages 630–631, 2005.
18 Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *ICALP (2)*, pages 508–520, 2009.
19 Adam Meyerson. Online facility location. In *FOCS*, pages 426–431, 2001.
20 Adam Meyerson, Kamesh Munagala, and Serge A. Plotkin. Designing networks incrementally. In *FOCS*, pages 406–415, 2001.
21 Chaitanya Swamy and David B. Shmoys. Approximation algorithms for 2-stage stochastic optimization problems. *SIGACT News*, 37(1):33–46, 2006.
22 Wikipedia. Dirac delta function — Wikipedia, the free encyclopedia.
23 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *FOCS*, pages 222–227, 1977.