Fighting Uncertainty with Randomness

An Introduction to Online Algorithms

Arjun Agarwal, BSc III

CMI Student Seminars Fall 2026

Contents

Introduction

Ice Skating Problem

Let's say a person wants to take some ice skating lessons.

- They could take an individual class for $1,
- a three month membership for $40
- a six month membership for $60,
- an year long membership for $75.

Every month has say 20 classes.

So what do we do?

What are algorithms?

Two undeniably true things about the human experience:

1. We are absolutely surrounded by problems
2. Our time and resources are far too limited to spend forever solving them

This is the entire motivation for the study of algorithms.

Long before computers existed, people developed algorithms, **literally a series of steps to solve a problem**, for arithmetic, navigation, voting, logistics and trade among variety of topics.

Computers simply gave us the ability to apply those ideas at otherwise impossible scales.

Online Algorithms

While in the classical setting, an algorithm is given all the information upfront. For example:

- a route finding algorithm usually takes the whole map as the input before computing the shortest route.
- a voting algorithm takes all the votes as input before telling us the winner(s).
- a packing algorithm takes the size of goods and space in ships as input before giving us an optimal packing scheme.

But in many real-world settings, we have to make decisions under uncertainty.

This uncertainty might stem from incomplete information, changing environments or unpredictable events.

This subfield of algorithms, named **Online Algorithms** is what we will talk about today.

Ski-Rental Problem

A Simpler Problem

Let's talk about a slightly simpler problem.

Ski-Rental Problem

A person is planning to try skiing out. They can rent skis out for \$1 and buy a pair for \$B.

We will use $B = 10$ as a running example.

Let's say the person is very self-aware and knows exactly how many times they will go skiing. If that number is less than 10, they are better off renting and if it is greater, they are better off buying. Easy!

However, most of us don't really posses the ability to look deep inside our souls and find the answers to 'how many times will I ski?'. Hence, the above method is not very practical and is often called 'the Prophet method'.

A slightly more practical but naive method is to buy a pair the 10th time you go to Ski. If you get bored before the 10th ski, you spent as little as possible. On the other hand, if you end up skiing more than 10 times, you spent \$19 over a 'Prophet' spending \$10. That is almost twice as much. Could we do better?

Making the purchase at a different day $\alpha \in [0, 10]$ doesn't seem to work any better as well as if we end up skating α days, we have overpaid by a ratio of

$$\frac{10 + \alpha - 1}{\alpha} = \frac{9}{\alpha} + 1 \leq 1.9$$

And for a general B, the competitive ratio goes to 2.

Flipping ~~the table~~ some coins

We created a bad bound for our algorithm by choosing a specific value of the number of days. While we can't stop the universe from being mean, we can be a little random and make it harder for us to be caged into a worst case.

Consider this scheme: On the 8th ski trip, you flip a coin. If it turns up heads, you buy skis now otherwise you buy it on the 10th trip. This performs better on average.

- If you call it quits before the 8th trip, it's the same as 'Prophet'.
- If you quit after the 8th trip, you spend $\frac{8+(10+7)}{2} = 12.5$ in comparison to the Prophet's 8 for a ratio of 1.56.
- If you quit after the 9th trip, you spend $\frac{9+17}{2} = 13$ in comparison to a Prophet's 9 for a ratio of 1.44.
- If you make more than 10 trips, you spend $\frac{19+17}{2} = 18$ in comparison to a Prophet's 10 for a ratio of 1.8.

Flipping ~~the table~~ some coins

Thus, even in the worst case, this is better than our break-even strategy. The worst possible ratio against the prophet is called the 'Competitive Ratio'. In this case, it was 1.8.

Can we do better?

As it turns out, yes. Instead of flipping an even coin on visit 8, let's flip a special coin that turn's up head 2/3 times and tails 1/3 times. This will give a ratio of about 1.76.

Actually, there is nothing special about visit 8 and visit 10. We could roll a special 10 sided dice and buy on the day that turns up on the dice. The dice is surely not even, as we should be much more likely to buy on the 10th visit than on the 1st.

An equivalent form is that we have a random variable A with support on $[1, 10]$ that we observe before the first visit and buy after those many visits.

So how do we choose the probabilities of A taking these values?

Towards a solution

Assume that the universe hates you and suppose our strategy has the worst expected performance if you stop ski-ing on day 6. In that case, universe would always make us stop ski-ing at day 6.

But then you can modify the probabilities of our dice with that information, but so can the universe (by moving to some other day) and so can you and so on...

This would mean, the universe really doesn't have a preference on which day to make you stop skiing. That is, the expected ratio is same for each of the 10 days (note, the algorithms performance doesn't change post the 10th day as the 'prophet' spends $10 while our expected spend is also same as day 10).

A Set of Linear Equations

Let's assume the ratio to be r. We can now make a linear equation in 10 variables

$$\begin{aligned} 10p_1 + p_2 + p_3 + ... + p_{10} &= r \\ 10p_1 + 11p_2 + 2p_3 + ... + 2p_{10} &= 2r \\ 10p_1 + 11p_2 + 12p_3 + ... + 3p_{10} &= 3r \\ &\vdots \\ 10p_1 + 11p_2 + 12p_3 + ... + 19p_{10} &= 10r \end{aligned}$$

with $p_i = \mathbb{P}(A = i)$. We also need to enforce $p_1 + p_2 + ... + p_{10} = 1$ and $0 \leq p_i \leq 1$ for all i.

Solving the Equations

With these, the only feasible solution is found at $r = \frac{1}{1-\left(1-\frac{1}{10}\right)^{10}} \approx 1.53533993279$ with $p_i = \frac{r}{10}\left(\frac{11}{10}\right)^{i-1}$.

As the number of days and cost increase, this bound converges to $r = \frac{3}{e-1} \approx 1.58197670687$ and $p_i = \frac{r}{n}\left(1+\frac{1}{n}\right)^{i+1}$ for $1 \leq i \leq n$ as $n \to \infty$.

That means, even if the universe hates us, on the long run, we don't overpay by more than 60 cents on the dollar. But could we do better?

Yao's Principle

Yao's Principle

In randomized algorithms, a common tool used to prove optimality is something called Yao's Principle:

Yao's Principle

For a given problem where the 'cost' for input instance x and algorithm a is $c(x, a)$, $\mathcal{X}$ is a distribution over the space of algorithms and $\mathcal{A}$ is a distribution over the algorithms to solve the problem:

$$\max_{x \in \mathcal{X}} \mathbb{E}(c(x, \mathcal{A})) \geq \min_{a \in \mathcal{A}} \mathbb{E}(c(\mathcal{X}, a))$$

The left-hand side of the inequality is what will will try to lower-bound: It is the worst-case performance of randomized algorithm $\mathcal{A}$. The right-hand side will be easier to talk about, because algorithms there are deterministic.

The proof is pretty much just moving of summations.

Optimality for Ski-Rental Problem

For the Ski-Rental Problem, note that any deterministic algorithm can be characterized by the day a on which it buys.

If $\mathcal{X} \leq a$, we incur cost $\mathcal{X}$ while we incur the cost $a + B$ otherwise.

This makes $\min_{a \in \mathcal{A}} \mathbb{E}(c(\mathcal{X}, a)) = \min_{a \in \mathcal{A}} \mathbb{E}(\mathcal{X} \mid \mathcal{X} \leq a)\mathbb{P}(\mathcal{X} \leq a) + (a + B)\mathbb{P}(\mathcal{X} > a)$

Let $S_i = \mathbb{P}(\mathcal{X} \geq i))$.

Thus, we can rewrite as: $\min_{a \in \mathcal{A}} \mathbb{E}(c(\mathcal{X}, a)) = \sum_{i=1}^{a} S_i + BS_{a+1}$.

We would like to choose a distribution such that the choice of a doesn't matter as if it did, we could choose a different $\mathcal{X}$ for a worse bond.

As $\mathbb{E}(c(\mathcal{X}, a+1)) = \mathbb{E}(c(\mathcal{X}, a)) \Rightarrow S_{a+1} = \left(1 - \frac{1}{B}\right) S_a$

We can use $S_1 = 1$ and thus, $S_i = \left(1 - \frac{1}{B}\right)^{i-1}$. This is a geometric distribution with parameter $\frac{1}{B}$, truncated at B.

Optimality for Ski-Rental Problem

Thus, we can't have better than a $\frac{e}{e-1}$ competitive algorithm which we achieved above.

Consider the following problem

Button Problem

There are m buttons and pressing the button i has the cost b_i. Let $b_1 \leq b_2 \leq \ldots \leq b_m$. Buttons beyond an index $1 \leq j \leq m$ are called 'target'. Pressing a 'target' button ends the game.

We want to minimize the cost to end the game.

Two Observations

An algorithm for the Button Problem can be characterized by a series of buttons $s_1, s_2, ..., s_k$ we press.

1. For an ideal algorithm, $s_i > s_j \iff i > j$ or the fact that an ideal algorithm doesn't backtrack.
2. $s_i = i$ as if any of the index (which has non-zero probability of appearing) is missed, say $m < s_j$, we can force the algorithm to pay $b_{s_1} + ... + b_{s_j} > b_{s_j}$ in place of b_m. This could be arbitrarily bad.

$$\min_{a \in \mathcal{A}} \mathbb{E}(c(\mathcal{X}, a))$$

$$= \min_{(s_1, s_2, \ldots, s_k)} \sum_{i=1}^{k} \mathbb{P}(\mathcal{X} > s_{i-1}) b_{s_i} \qquad \text{using observation 1}$$

Consider the distribution (that was an arbitrary choice we have!)

$$\mathbb{P}(\mathcal{X} = j) = \frac{1}{e}\left(1 - \frac{1}{e}\right)^{j-1} \Rightarrow \mathbb{P}(\mathcal{X} > j) = \left(1 - \frac{1}{e}\right)^{j}$$

.

Then, our expression is:

$$= \min_{(s_1, s_2, \ldots, s_k)} \sum_{i=1}^{k} \left(1 - \frac{1}{e}\right)^{s_{i-1}} b_{s_i}$$

$$= \sum_{i=1}^{m} \left(1 - \frac{1}{e}\right)^{i-1} b_i \qquad \text{using observation 2}$$

Notice, the expected optimal cost is

$$\sum_{i=1}^{m} b_i \mathbb{P}(\mathcal{X} = i)$$

$$= \sum_{i=1}^{m} b_i \frac{1}{e} \left(1 - \frac{1}{e}\right)^{i-1}$$

$$= \frac{1}{e} \sum_{i=1}^{m} b_i \left(1 - \frac{1}{e}\right)^{i-1}$$

which is e times the OPT.

Thus, we can't have a better than e competitive online algorithm for this problem.

Back To Ice Skating

We will now formalize our initial problem.

Ice Skating Problem

A person is trying our Ice Skating and the option to buy membership for $p_1, p_2, \ldots, p_n \in \mathbb{Z} \cup \{\infty\}$ at cost $c_1, c_2, \ldots, c_n \in \mathbb{R}$ respectively.

Further assume, $p_i > p_j \iff c_i > c_j$.

If we reflected on our deepest wants and needs and our true personality and figured that we want to ice-skate for some t days, we could simply use the fact that:

$$\mathrm{OPT}(t) = \min_i (c_i + \mathrm{OPT}(t - p_i))$$

Note, the above can be solved quite quickly using standard dynamic programming.

We will also point out that the function $\mathrm{best}(b) =$
maximum number of days one can purchase in cost b can be computed using a similar recurrence

$$\mathrm{best}(b) = \max_i (p_i + \mathrm{best}(b - c_i))$$

A Simple Deterministic Algorithm

A classic idea is online algorithms is doubling the budget as we go on (we sort of did the same thing with buying the moment cumulative cost had exceeded in Ski-Rental). Consider:

```
for each round i = 0, 1, ...
  set budget to 2^i
  append best(budget) to solution
```

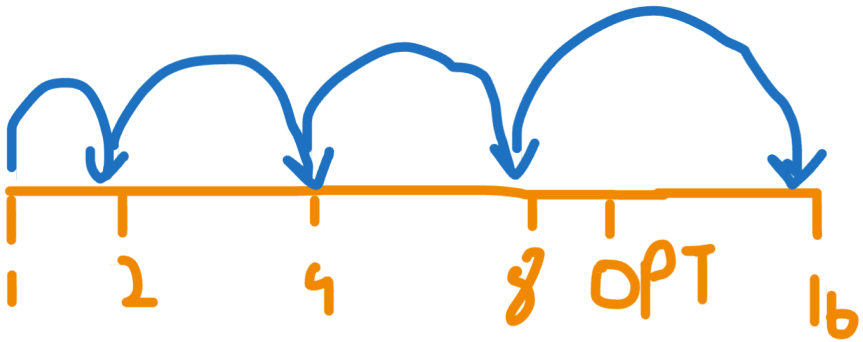
We claim this is 4-competitive.

Assume that $2^i < \mathrm{OPT}(t) < 2^{i+1}$. Then in the $i+1$-th iteration, we will definitely be done. And in the worst case, the previous rounds wouldn't have covered t. Thus,

$$\mathrm{SOL}(t) \leq 2^{i+1} + 2^i + \ldots + 1 = 2^{i+2} - 1$$
$$\Rightarrow \mathrm{SOL}(t) \leq 4\mathrm{OPT}(t)$$

Thus, the competitive ratio is atmost 4.

We can show that this is sharp by the construction:



Suppose we didn't choose our starting point at 1 and instead as 1.125. Then, we would reach 9, avoid the overshooting to 16 and end up with a ratio of $\frac{15}{8} = 1.875$.

But the universe could then perhaps choose a different value (say 10) to screw the ratio. Perhaps it is time that we bring out randomness?

Suppose our initial budget is $\alpha \in [1, 2]$ chosen uniformly.

```
choose α uniformly from [1,2]
for each round i = 0, 1, ...
  set budget to α * 2^i
  append best(budget) to solution
```

What is the ratio now?

Let $\mathrm{OPT}(t) = 2^k x$ with $x \in [1, 2]$.

For $\alpha \geq x$, our algorithm terminates in atmost k rounds. The ratio is bounded by $\frac{\alpha(1+2+\ldots+2^k)}{2^k x} \leq \frac{2\alpha}{x}$.

For $\alpha < x$, our algorithm terminates in atmost $k+1$ rounds. The ratio is bounded by $\frac{\alpha(1+2+\ldots+2^{k+1})}{2^k x} \leq \frac{4\alpha}{x}$.

This would make the expected ratio:

$$\frac{1}{x}\left(\int_1^x 4\alpha d\alpha + \int_x^2 2\alpha d\alpha\right)$$

$$= \frac{1}{x}[2(x^2-1) + (4-x^2)]$$

$$= x + \frac{2}{x}$$

Which is bounded by 3 for valid values of x.

Exploiting Multiplicativity

But could we do better? Well, we should not be choosing α uniformly from $[1, 2]$ if our goal is to be as close to the unknown OPT in a multiplicative sense. It is sort of clearer by example, consider:

```
choose α uniformly from [0,1]
for each round i = 0, 1, ...
  set budget to 2^(α+i)
  append best(budget) to solution
```

Exploiting Multiplicativity

Let $\mathrm{OPT}(t) = 2^{k+x}$ with $x \in (0, 1)$.

We have the initial budget 2^{α} with α chosen uniformly from $[0, 1]$. Thus, our ratio is:

If $\alpha \geq x$, our algorithm terminates in atmost k rounds. The ratio is bounded by $2 \cdot 2^{\alpha - x}$.

If $\alpha < x$, our algorithm terminates in atmost $k + 1$ rounds. The ratio is bounded by $4 \cdot 2^{\alpha - x}$.

This would make the expected ratio:

$$
\begin{aligned}
&\frac{1}{2^x}\left(\int_0^x 4\cdot 2^\alpha d\alpha + \int_x^1 2\cdot 2^\alpha d\alpha\right)\\
&= \frac{1}{2^x}\left(\frac{4}{\ln 2}(2^x-1) + \frac{2}{\ln 2}(2-2^x)\right)\\
&= \frac{2}{\ln(2)}\\
&\approx 2.885
\end{aligned}
$$

Why Double?

However, one final thing worth noting: there is nothing sacred about doubling.

Suppose we scaled the budget by some c. That would give the expected ratio:

$$\begin{aligned}
&\frac{1}{2^x}\left(\int_0^x c^2 \cdot c^\alpha d\alpha + \int_x^1 c \cdot c^\alpha d\alpha\right) \\
&= \frac{1}{2^x}\left(\frac{c^2}{\ln c}(c^x - 1) + \frac{c}{\ln c}(c - c^x)\right) \\
&= \frac{c}{\ln(c)}
\end{aligned}$$

Which is minimized for $c = e$, giving the competitive ratio e.

Why Double?

This gives us the e competitive algorithm:

```
choose α uniformly from [0,1]
for each round i = 0, 1, ...
  set budget to e^(α+i)
  append best(budget) to solution
```

But could we do better? We claim that it is not possible.

Online Reductions

When Yao is too hard...

Using the Yao's Principle seems very hard in this case. So what do we do? Well, we can borrow the optimality from a different problem via an online reduction.

Online Reduction

We have an online reduction of P to Q if for any algorithm $\mathcal{A}$ giving an online competitive ratio of γ for Q, there exists an algorithm $\mathcal{A}'$ giving an online competitive ratio of γ for P which makes decisions without using any future information.

Symbolically, $\text{OPT}_Q(t) < \gamma \text{SOL}_{Q,\mathcal{A}}(t) \implies \exists \mathcal{A}'$ s.t. $\text{OPT}_P(t) < \gamma \text{SOL}_{P,\mathcal{A}'}(t) + \beta$

As we want to show e is optimal, we would perhaps like another problem with optimal ratio of e.

Given an instance of the button problem $b_1 \leq b_2 \leq \ldots \leq b_m$ and j, consider the Ice-Skating problem $(1, b_m), (2, b_m^2), \ldots, \left(b_m, b_m^{b_m}\right)$ and b_m^j.

If an algorithm buys membership i in the ice-skating instance, we press the button with the maximum cost less than i. If the clicked button turns out to be a target button, we report to $\mathcal{A}$ that the last skiing day has been reached and terminate the whole algorithm.

However, we might never be able to click a target button with only this procedure since the algorithm may choose only "cheap" options. In order to ensure that the algorithm always terminates, we add the following condition. Once the total cost incurred becomes at least b_m, we click the last button m and terminate.

Other than the last case (which is absorbed by the additive factor), notice that our algorithms work with same ratios. Thus, e is optimal.

Conclusion

The field of online algorithms is a natural extension to the classical study of algorithms and had been considered since antiquity. After all, Uncertainty has belonged to every age, every moment.

In 1985, Sleator and Tarjan introduced the idea of competitive analysis to talk about caching.

The **Ski-Rental Problem** was first solved in 1986 by Karlin, Manasse, Rudolph and Sleator under the name **'snoopy catching'** which is another hardware problem.

The **Ski-Rental Problem** was introduced as a pedological tool in the 1994 paper "Competitive randomized algorithms for non-uniform problems" by Karlin, Manasse, McGeoch and Owicki.

The **Ice Skating Problem** is a natural generalization of the Ski-Rental Problem and is usually studied under the name **Multi-Option Ski Rental Problem**.

The deterministic algorithm was given by Guiqing Zhang, Chung Keung Poon, Yinfeng Xu in 2011.

The optimal randomized algorithm was given by Yongho Shin, Changyeol Lee, Gukryeol Lee, Hyung-Chan An in 2023.

While I do use the same algorithms and achive the same bounds, most of the proofs were re-worked or rederived for pedological reasons.

I hope the talk made you think of some online problems you might have encountered (and perhaps you already have some idea on trying to solve them!).

Thank You!
Any Questions?

References

- *Shin, Y., Lee, C., Lee, G., & An, H.-C. (2023).* **Improved Learning-Augmented Algorithms for the Multi-Option Ski Rental Problem via Best-Possible Competitive Analysis.** Proceedings of the 40th International Conference on Machine Learning (ICML), 31539–31561.
- *Zhang, G., Poon, C. K., & Xu, Y. (2011).* **The Ski-Rental Problem with Multiple Discount Options.** Information Processing Letters.
- *Molinaro, M., & Fernandes, M. L. S.* **Lecture 1: Online Algorithms.**
- *Singla, S.* **Lecture 8: Hardness: Yao's Minimax Lemma.**
- *Yao's Principle.* **Wikipedia**, 2026.