Fun of Online Algorithms

Enrichment Lecture

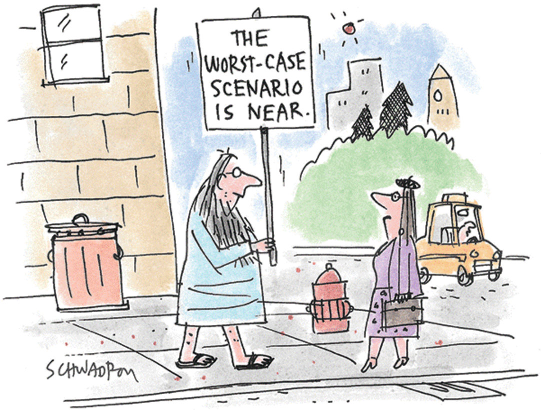
Sid Chi-Kin Chau

Australian National University

sid.chau@anu.edu.au

November 5, 2022

Uncertain future? Help! 😱

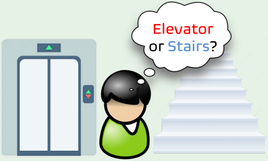


- There is always uncertainty in future. How to cope with uncertain future?

Online Decision Problem #1: Elevator or Stairs ⏳

Elevator or Stairs

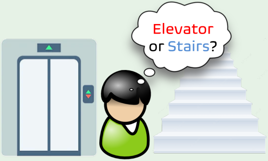
- You can either use the elevator but need to wait, or take the stairs
- It takes E mins to get to your floor by elevator (once it comes)
- But the waiting time for elevator is unknown
- It takes S mins by stairs, where $S > E$



Online Decision Problem #1: Elevator or Stairs ⏳

Elevator or Stairs

- You can either use the elevator but need to wait, or take the stairs
- It takes E mins to get to your floor by elevator (once it comes)
- But the waiting time for elevator is unknown
- It takes S mins by stairs, where $S > E$

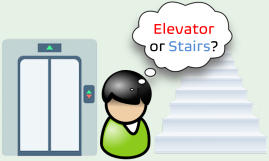


- If you are an oracle 🔮 (i.e., you can predict the future), what will you do?

Online Decision Problem #1: Elevator or Stairs ⏳

Elevator or Stairs

- You can either use the elevator but need to wait, or take the stairs
- It takes E mins to get to your floor by elevator (once it comes)
- But the waiting time for elevator is unknown
- It takes S mins by stairs, where $S > E$

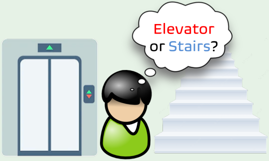


- If you are an oracle 🔮 (i.e., you can predict the future), what will you do?
- If the waiting time $W \geq S - E$, then take the stairs, else wait for the elevator
- This is called the **offline optimal solution**

Online Decision Problem #1: Elevator or Stairs ⏳

Elevator or Stairs

- You can either use the elevator but need to wait, or take the stairs
- It takes E mins to get to your floor by elevator (once it comes)
- But the waiting time for elevator is unknown
- It takes S mins by stairs, where $S > E$

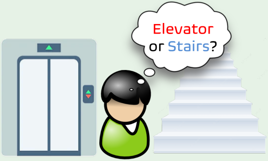


- If you are an oracle 🔮 (i.e., you can predict the future), what will you do?
- If the waiting time $W \geq S - E$, then take the stairs, else wait for the elevator
- This is called the **offline optimal solution**
- But you are not an oracle, you don't know the true W, how long should you wait for the elevator?

Online Decision Problem #1: Elevator or Stairs ⏳

Elevator or Stairs

- You can either use the elevator but need to wait, or take the stairs
- It takes E mins to get to your floor by elevator (once it comes)
- But the waiting time for elevator is unknown
- It takes S mins by stairs, where $S > E$

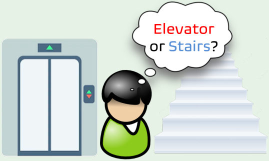


- If you are an oracle 🔮 (i.e., you can predict the future), what will you do?
- If the waiting time $W \geq S - E$, then take the stairs, else wait for the elevator
- This is called the **offline optimal solution**
- But you are not an oracle, you don't know the true W, how long should you wait for the elevator? This is called an **online decision problem**

Online Decision Problem #1: Elevator or Stairs ⏳

Elevator or Stairs

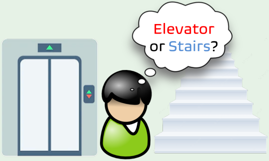
- You can either use the elevator and wait, or take the stairs
- It takes E mins to get to your floor by elevator (once it comes)
- But the waiting time for elevator is unknown
- It takes S mins by stairs



Online Decision Problem #1: Elevator or Stairs ⏳

Elevator or Stairs

- You can either use the elevator and wait, or take the stairs
- It takes E mins to get to your floor by elevator (once it comes)
- But the waiting time for elevator is unknown
- It takes S mins by stairs



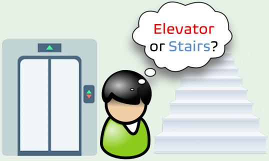
Waiting Strategy

- Wait at most X mins for the elevator
- If elevator does not come in X mins, then take the stairs

Online Decision Problem #1: Elevator or Stairs ⏳

Elevator or Stairs

- You can either use the elevator and wait, or take the stairs
- It takes E mins to get to your floor by elevator (once it comes)
- But the waiting time for elevator is unknown
- It takes S mins by stairs



Waiting Strategy

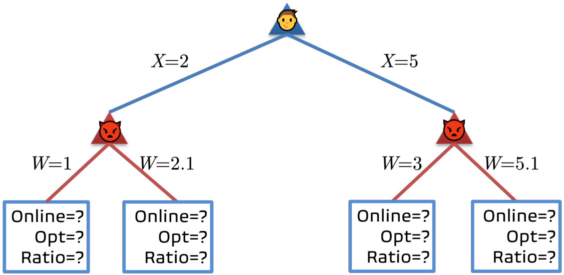
- Wait at most X mins for the elevator
- If elevator does not come in X mins, then take the stairs

- How do we choose X?
- Is there a way to systematically analyze the waiting strategy with respect to X?

⋘Imagine Playing Chess with Adversary⋙

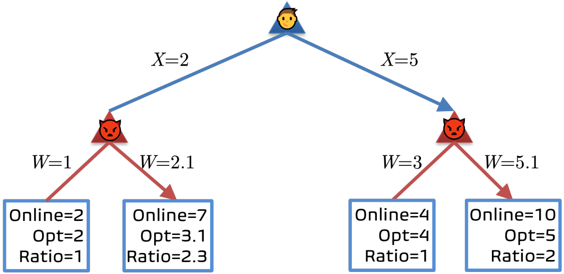
Playing a Zero-sum Game with Adversary

- Suppose $E = 1$ and $S = 5$
- Let the offline optimal total time be Opt, and the online total time be Online
- Let the ratio be $\frac{\text{Online}}{\text{Opt}}$
- You can choose X either $X = 2$ or $X = 5$



Playing a Zero-sum Game with Adversary

- Suppose $E = 1$ and $S = 5$
- Let the offline optimal total time be Opt, and the online total time be Online
- Let the ratio be $\frac{\text{Online}}{\text{Opt}}$
- You can choose X either $X = 2$ or $X = 5$
- In general, what X will you choose to minimize the ratio between Online and Opt?



Elevator or Stairs ⏳

Waiting Strategy $\mathcal{A}_{\mathsf{EoS}}(X)$

- Wait at most X mins for the elevator
- If elevator does not come in X mins, then take the stairs

Elevator or Stairs ⏳

Waiting Strategy $\mathcal{A}_{\mathsf{EoS}}(X)$

- Wait at most X mins for the elevator
- If elevator does not come in X mins, then take the stairs

- The total time of $\mathcal{A}_{\mathsf{EoS}}(X) = \begin{cases} W + E, & \text{if } W \leq X \\ X + S, & \text{if } W > X \end{cases}$

Elevator or Stairs ⏳

Waiting Strategy $\mathcal{A}_{\text{EoS}}(X)$

- Wait at most X mins for the elevator
- If elevator does not come in X mins, then take the stairs

- The total time of $\mathcal{A}_{\text{EoS}}(X) = \begin{cases} W + E, & \text{if } W \leq X \\ X + S, & \text{if } W > X \end{cases}$
- Let the ratio: $\mathtt{R}(W, X) = \begin{cases} \frac{W+E}{\min\{S, W+E\}}, & \text{if } W \leq X \\ \frac{X+S}{\min\{S, W+E\}}, & \text{if } W > X \end{cases}$

Elevator or Stairs ⌛

Waiting Strategy $\mathcal{A}_{\mathsf{EoS}}(X)$

- Wait at most X mins for the elevator
- If elevator does not come in X mins, then take the stairs

- The total time of $\mathcal{A}_{\mathsf{EoS}}(X) = \begin{cases} W + E, & \text{if } W \leq X \\ X + S, & \text{if } W > X \end{cases}$
- Let the ratio: $\mathtt{R}(W, X) = \begin{cases} \frac{W+E}{\min\{S, W+E\}}, & \text{if } W \leq X \\ \frac{X+S}{\min\{S, W+E\}}, & \text{if } W > X \end{cases}$
- How will Adversary choose W? How will you choose X in response?

Elevator or Stairs ⏳

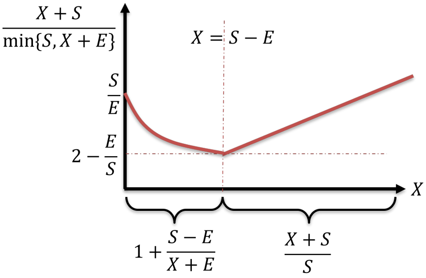
Waiting Strategy $\mathcal{A}_{\mathsf{EoS}}(X)$

- Wait at most X mins for the elevator
- If elevator does not come in X mins, then take the stairs

- The total time of $\mathcal{A}_{\mathsf{EoS}}(X) = \begin{cases} W+E, & \text{if } W \le X \\ X+S, & \text{if } W > X \end{cases}$
- Let the ratio: $\mathtt{R}(W,X) = \begin{cases} \frac{W+E}{\min\{S,W+E\}}, & \text{if } W \le X \\ \frac{X+S}{\min\{S,W+E\}}, & \text{if } W > X \end{cases}$
- How will Adversary choose W? How will you choose X in response?
- Let the **competitive ratio** be:

$$\alpha(\mathcal{A}_{\mathsf{EoS}}) = \min_X \max_W \ \mathtt{R}(W,X)$$

Elevator or Stairs ⏳



- The competitive ratio of $\mathcal{A}_{\mathsf{EoS}}$ is obtained by optimizing X and assuming Adversary maximizes $\mathtt{R}(W, X)$ by choosing $W = X + \epsilon$ for a very small positive ϵ:

$$\alpha(\mathcal{A}_{\mathsf{EoS}}) = \min_X \max_W \ \mathtt{R}(W, X) = \min_X \ \frac{X+S}{\min\{S, X+E\}} = 2 - \frac{E}{S}$$

where the minimum value is at $X = S - E$

Elevator or Stairs ⏳

Theorem

$\mathcal{A}_{\mathrm{EoS}}(X)$ *has the best possible competitive ratio for any online algorithms*

Proof:

- Every online algorithm can be expressed as a strategy played on a zero-sum game against Adversary
- Suppose the maximum time that an online algorithm stops waiting is X

Elevator or Stairs ⏳

Theorem

$\mathcal{A}_{\mathsf{EoS}}(X)$ *has the best possible competitive ratio for any online algorithms*

Proof:

- Every online algorithm can be expressed as a strategy played on a zero-sum game against Adversary
- Suppose the maximum time that an online algorithm stops waiting is X
- Every online algorithm can be expressed by $\mathcal{A}_{\mathsf{EoS}}(X)$
- Then the competitive ratio of every online algorithm is lower bounded by:

$$\min_{X} \max_{W} \mathtt{R}(W, X) \geq \min_{X} \alpha(\mathcal{A}_{\mathsf{EoS}}(X))$$

Online Decision Problems

- *Online decision problems*:
  - Problems are *not* always solved in one shot, but progressively and continually
  - Consider this decision problem
    - ★ Input is revealed gradually as time evolves
    - ★ A decision has to be made from time to time, given partial input
    - ★ But an optimal decision depends on all future input (so cannot make optimal decision)
    - ★ Decisions made cannot be retracted
  - Examples:
    - ★ How much should a student learn to pass an exam?
    - ★ When should we sell/buy in stock markets?
    - ★ How to find your true love?

Online Decision Problems

- *Online decision problems*:
  - Problems are *not* always solved in one shot, but progressively and continually
  - Consider this decision problem
    - Input is revealed gradually as time evolves
    - A decision has to be made from time to time, given partial input
    - But an optimal decision depends on all future input (so cannot make optimal decision)
    - Decisions made cannot be retracted
  - Examples:
    - How much should a student learn to pass an exam?
    - When should we sell/buy in stock markets?
    - How to find your true love?
- *Online algorithms*:
  - Solve online decision problems without knowing the entire input from the start to the end
- Motto: *Always prepare for the worst-case scenario; if it is the best you'll win anyway*

⋘*Let's Apply Online Algorithms to Financial Market Trading*⋙

Online Decision Problem #2: Trading in Financial Markets

- When should you sell/buy in stock markets without knowing the future of market prices?
- Trading in financial market is an online decision problem
  - Goal: Want to sell at the highest price, or buy at the lowest price
- Decisions need to be made without complete future information
- How do we know if the current price is the highest/lowest?
- Probabilistic analysis
  - Need to model risk and uncertainty of future price fluctuations
- Competitive online algorithms
  - Risk-less, guaranteeing the worst case performance

1-Max Search

Definition (1-Max Search)

- Give a sequence of prices $(p_1, p_2, ..., p_T)$ over time
- Goal: Decide whether to sell at price p_t at current time t,
  or wait for the next time at an unknown price
- Unknown:
  - Assume no knowledge of future price p_t
- Known:
  - Price range $m \leq p_t \leq M$ for all $t = 1, ..., T$
  - Deadline: If not sold before T, then will be forced to sell at p_T

1-Max Search

Definition (1-Max Search)

- Give a sequence of prices $(p_1, p_2, ..., p_T)$ over time
- Goal: Decide whether to sell at price p_t at current time t,
  or wait for the next time at an unknown price
- Unknown:
  - Assume no knowledge of future price p_t
- Known:
  - Price range $m \leq p_t \leq M$ for all $t = 1, ..., T$
  - Deadline: If not sold before T, then will be forced to sell at p_T

- Offline optimal solution:
  - Pick the time to sell at the highest price: $t_{\max} = \arg\max\{p_t : t = 1, ..., T\}$
- Online algorithm: How?

1-Max Search

Threshold Selling Algorithm $\mathcal{A}_{\mathsf{1max}}(\hat{p})$

- Repeat
  - If the current price $p_t \geq \hat{p}$, then sell and exit
  - Else wait for the next price p_{t+1}
- Until $t = T$ then sell at p_T

1-Max Search

Threshold Selling Algorithm $\mathcal{A}_{\text{1max}}(\hat{p})$

- Repeat
  - If the current price $p_t \geq \hat{p}$, then sell and exit
  - Else wait for the next price p_{t+1}
- Until $t = T$ then sell at p_T

Lemma

Setting $\hat{p} = \sqrt{Mm}$ in $\mathcal{A}_{\text{1max}}(\hat{p})$ achieves the best competitive ratio $= \sqrt{\frac{M}{m}}$

Proof:

1-Max Search

Threshold Selling Algorithm $\mathcal{A}_{\text{1max}}(\hat{p})$

- Repeat
  - If the current price $p_t \geq \hat{p}$, then sell and exit
  - Else wait for the next price p_{t+1}
- Until $t = T$ then sell at p_T

Lemma

Setting $\hat{p} = \sqrt{Mm}$ in $\mathcal{A}_{\text{1max}}(\hat{p})$ achieves the best competitive ratio $= \sqrt{\frac{M}{m}}$

Proof:

- Knowing $\hat{p}$, Adversary has two options:
  1. Case 1: Make $\mathcal{A}_{\text{1max}}$ sell at $\hat{p}$
  2. Case 2: Make $\mathcal{A}_{\text{1max}}$ sell at p_T

1-Max Search

Proof:

- Knowing $\hat{p}$, Adversary has two options:
  1. Case 1: Make $\mathcal{A}_{\text{1max}}$ sell at $\hat{p}$
  2. Case 2: Make $\mathcal{A}_{\text{1max}}$ sell at p_T

1-Max Search

Proof:

- Knowing $\hat{p}$, Adversary has two options:
  1. Case 1: Make $\mathcal{A}_{\text{1max}}$ sell at $\hat{p}$
  2. Case 2: Make $\mathcal{A}_{\text{1max}}$ sell at p_T
- The competitive ratio is
  1. Case 1: $\frac{M}{\hat{p}}$ by price sequence $(\hat{p}, M, ...)$
  2. Case 2: $\frac{\hat{p}}{m}$ by price sequence $(\hat{p} - \epsilon, ..., m)$
- We optimize $\hat{p}$ by minimizing the following value:

$$\min_{\hat{p}} \ \max\{\frac{M}{\hat{p}}, \frac{\hat{p}}{m}\}$$

1-Max Search

Proof:

- Knowing $\hat{p}$, Adversary has two options:
  1. Case 1: Make $\mathcal{A}_{1\text{max}}$ sell at $\hat{p}$
  2. Case 2: Make $\mathcal{A}_{1\text{max}}$ sell at p_T
- The competitive ratio is
  1. Case 1: $\frac{M}{\hat{p}}$ by price sequence $(\hat{p}, M, ...)$
  2. Case 2: $\frac{\hat{p}}{m}$ by price sequence $(\hat{p} - \epsilon, ..., m)$
- We optimize $\hat{p}$ by minimizing the following value:

$$\min_{\hat{p}} \ \max\{\frac{M}{\hat{p}}, \frac{\hat{p}}{m}\}$$

- Note that $\frac{M}{\hat{p}}$ is decreasing in $\hat{p}$; $\frac{\hat{p}}{m}$ is increasing in $\hat{p}$

1-Max Search

Proof:

- Knowing $\hat{p}$, Adversary has two options:
  1. Case 1: Make $\mathcal{A}_{1\text{max}}$ sell at $\hat{p}$
  2. Case 2: Make $\mathcal{A}_{1\text{max}}$ sell at p_T
- The competitive ratio is
  1. Case 1: $\frac{M}{\hat{p}}$ by price sequence $(\hat{p}, M, ...)$
  2. Case 2: $\frac{\hat{p}}{m}$ by price sequence $(\hat{p} - \epsilon, ..., m)$
- We optimize $\hat{p}$ by minimizing the following value:

$$\min_{\hat{p}} \ \max\{\frac{M}{\hat{p}}, \frac{\hat{p}}{m}\}$$

- Note that $\frac{M}{\hat{p}}$ is decreasing in $\hat{p}$; $\frac{\hat{p}}{m}$ is increasing in $\hat{p}$
- The optimal setting: $\frac{M}{\hat{p}} = \frac{\hat{p}}{m} \ \Rightarrow \ \hat{p} = \sqrt{Mm}$

1-Min Search

- Need to buy at as low price as possible

Threshold Buying Algorithm $\mathcal{A}_{\text{1min}}(\hat{p})$

- Repeat
  - If the current price $p_t \leq \hat{p}$, then buy and exit
  - Else wait for the next price p_{t+1}
- Until $t = T$ then buy at p_T

1-Min Search

- Need to buy at as low price as possible

Threshold Buying Algorithm $\mathcal{A}_{\text{1min}}(\hat{p})$

- Repeat
  - If the current price $p_t \le \hat{p}$, then buy and exit
  - Else wait for the next price p_{t+1}
- Until $t = T$ then buy at p_T

Lemma

Setting $\hat{p} = \sqrt{Mm}$ *in* $\mathcal{A}_{\text{1min}}(\hat{p})$ *achieves the best competitive ratio* $= \sqrt{\frac{M}{m}}$

k-Max Search

Definition (k-Max Search)

- Goal: Need to sell k items, only one item sold at each time
- Known:
  - Price range $m \leq p_t \leq M$
  - Deadline: If i items unsold before $T - i + 1$, then will be forced to sell all at $(p_{T-i+1}, ..., p_T)$
- Offline optimal solution:
  - Pick the k highest prices
- Online algorithm: How?
- We extend from $\mathcal{A}_{\text{1max}}$ to $\mathcal{A}_{\text{kmax}}$ with k threshold selling prices

k-Min Search

Definition (k-Min Search)

- Goal: Need to buy k items, only one item bought at each time
- We extend from $\mathcal{A}_{\text{1min}}$ to $\mathcal{A}_{\text{kmin}}$ with k threshold buying prices
- Let $\hat{p}_i$ be the threshold buying price of the i-th item
- $\hat{p}_1 \geq \hat{p}_2 \geq \ldots \geq \hat{p}_k$

Online Decision Problem #3: Online Dating Problem

- Goal: Find your true love by dating a stream of candidates
- Suppose you estimate that you will have n dates over the time
  - Your true love will be the best person out of the n candidates

Online Decision Problem #3: Online Dating Problem 💘

- Goal: Find your true love by dating a stream of candidates
- Suppose you estimate that you will have n dates over the time
  - Your true love will be the best person out of the n candidates
- Rules:
  - You can only date one person at a time (no cheating allowed 🙅)
  - You have to decide whether either you will
    - ★ Marry the current candidate
    - ★ Or break up with the current candidate to date the next (unknown) candidate
  - Broken-up relationship can't be rekindled (need to move on from past relationships 💔)

Online Decision Problem #3: Online Dating Problem 💘

- Goal: Find your true love by dating a stream of candidates
- Suppose you estimate that you will have n dates over the time
  - Your true love will be the best person out of the n candidates
- Rules:
  - You can only date one person at a time (no cheating allowed 🙅)
  - You have to decide whether either you will
    - ★ Marry the current candidate
    - ★ Or break up with the current candidate to date the next (unknown) candidate
  - Broken-up relationship can't be rekindled (need to move on from past relationships 💔)
- Dating is definitely an "online" decision problem

Reference Materials

- A Course in "Advanced Algorithms"
  - `https://users.cecs.anu.edu.au/~sid.chau/teaching.html`