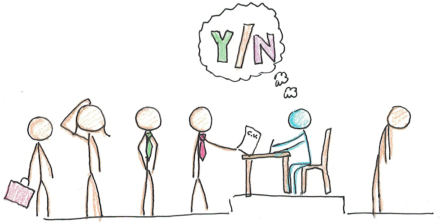
How to choose a life partner optimally?

Małgorzata Sulkowska

Université Côte d'Azur, Inria, France
Wrocław University of Science and Technology, Poland

C@FÉ ADSTIC
6 December 2021



Optimal stopping in theory

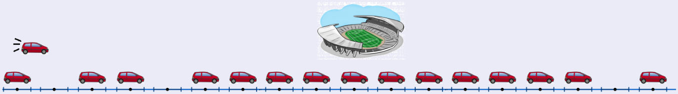
Problem of choosing a time to take a particular action in order to maximise gain or minimise cost.

Optimal stopping in theory

Problem of choosing a time to take a particular action in order to maximise gain or minimise cost.

Optimal stopping in practice

- Should I already park a car or drive further towards the goal?

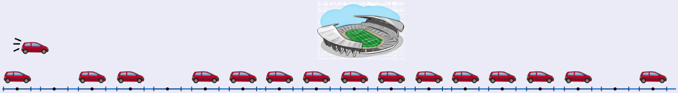


Optimal stopping in theory

Problem of choosing a time to take a particular action in order to maximise gain or minimise cost.

Optimal stopping in practice

- Should I already park a car or drive further towards the goal?



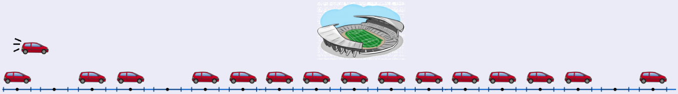
- Should I stop paying for fixing my car and buy a new one?

Optimal stopping in theory

Problem of choosing a time to take a particular action in order to maximise gain or minimise cost.

Optimal stopping in practice

- Should I already park a car or drive further towards the goal?



- Should I stop paying for fixing my car and buy a new one?

- Should I propose to my current partner?

Introduction to online algorithms in three steps

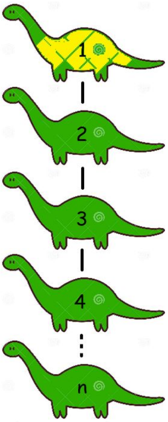
1. FIRST PROBLEM
2. SECOND PROBLEM

Introduction to online algorithms in three steps

1. SECRETARY PROBLEM
2. SECOND PROBLEM

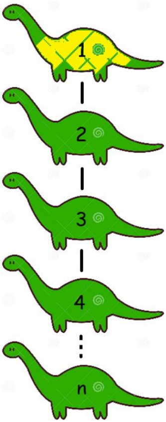
Secretary problem (marriage, dowry, ...), 1950s

- There are n linearly ordered candidates.
- They appear one by one in some random order.
- You know the relative ranks of the candidates met so far.



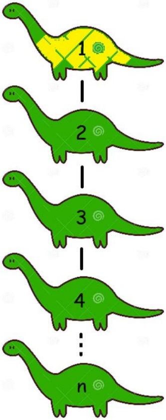
Secretary problem (marriage, dowry, ...), 1950s

- There are n linearly ordered candidates.
- They appear one by one in some random order.
- You know the relative ranks of the candidates met so far.
- You can accept or decline the current candidate.
- Declined candidate takes offence and never comes back.
- Accepted candidate is your choice.



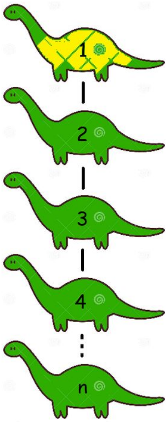
Secretary problem (marriage, dowry, ...), 1950s

- There are n linearly ordered candidates.
- They appear one by one in some random order.
- You know the relative ranks of the candidates met so far.
- You can accept or decline the current candidate.
- Declined candidate takes offence and never comes back.
- Accepted candidate is your choice.
- Your choice turns out to be 1: you win!! :) (gain 1).
- Your choice is not 1: you loose... :((gain 0).



Secretary problem (marriage, dowry, ...), 1950s

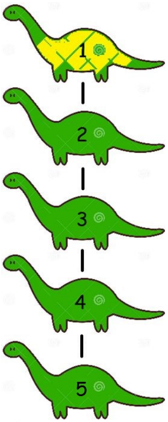
- There are n linearly ordered candidates.
- They appear one by one in some random order.
- You know the relative ranks of the candidates met so far.
- You can accept or decline the current candidate.
- Declined candidate takes offence and never comes back.
- Accepted candidate is your choice.
- Your choice turns out to be 1: you win!! :) (gain 1).
- Your choice is not 1: you loose... :((gain 0).



Aim: maximise the expected gain
(all permutations of candidates are equiprobable).
Equivalently: maximise the probability of choosing candidate no. 1.

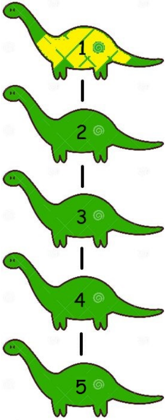
Let's play!

Random permutation: $\sigma = (5, 3, 4, 1, 2)$ (you do not know it!)



Let's play!

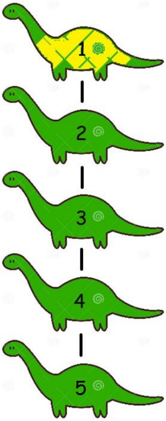
Random permutation: $\sigma = (5, 3, 4, 1, 2)$ (you do not know it!)

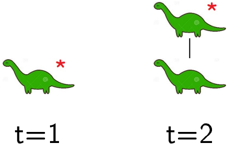


t=1

Let's play!

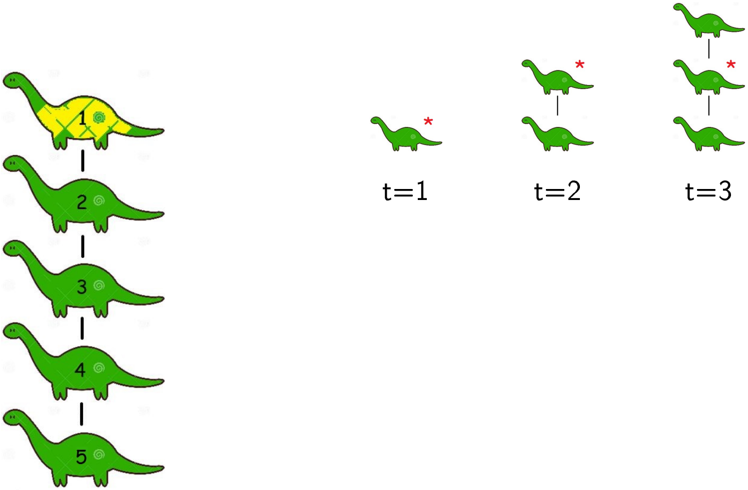
Random permutation: $\sigma = (5, 3, 4, 1, 2)$ (you do not know it!)





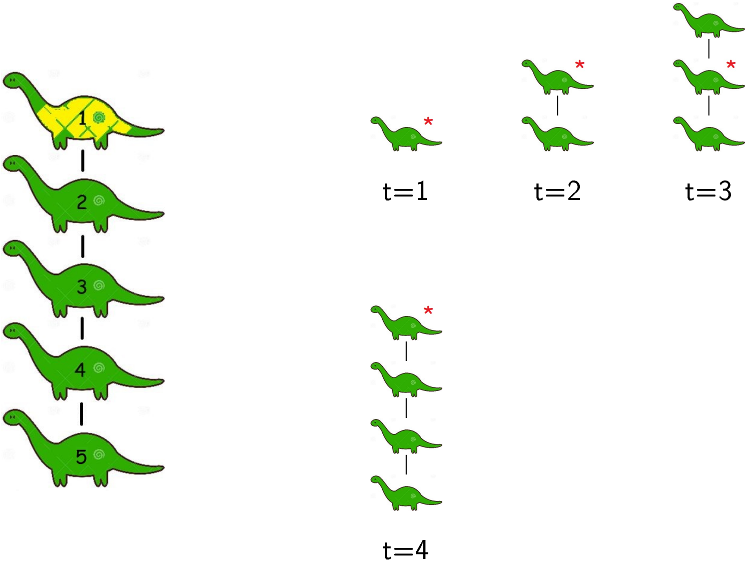
Let's play!

Random permutation: $\sigma = (5, 3, 4, 1, 2)$ (you do not know it!)



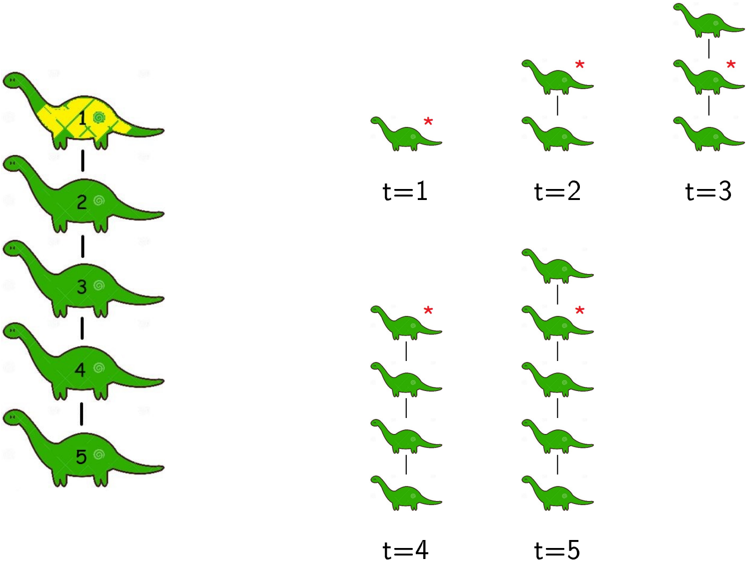
Let's play!

Random permutation: $\sigma = (5, 3, 4, 1, 2)$ (you do not know it!)



Let's play!

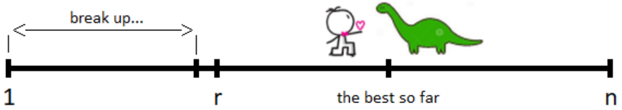
Random permutation: $\sigma = (5, 3, 4, 1, 2)$ (you do not know it!)



Secretary problem - an optimal solution

Optimal solution is of a threshold type:

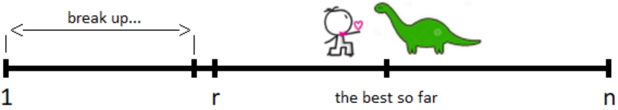
- wait until a certain threshold r;
- at this time or later accept the first candidate best so far.



Secretary problem - an optimal solution

Optimal solution is of a threshold type:

- wait until a certain threshold r;
- at this time or later accept the first candidate best so far.



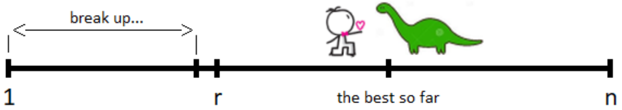
Probability of success:

$$\mathbb{P}[S] = \sum_{i=1}^{n} \mathbb{P}[S|\text{1 is at pos i}]\mathbb{P}[\text{1 is at pos i}]$$

Secretary problem - an optimal solution

Optimal solution is of a threshold type:

- wait until a certain threshold r;
- at this time or later accept the first candidate best so far.



Probability of success:

$$\mathbb{P}[S] = \sum_{i=1}^{n} \mathbb{P}[S|\text{1 is at pos i}]\mathbb{P}[\text{1 is at pos i}]$$

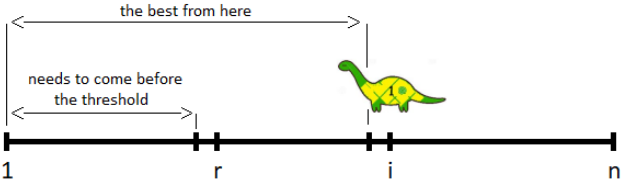
$$= \sum_{i=1}^{n} \mathbb{P}[S|\text{1 is at pos i}]\frac{1}{n}$$

Secretary problem - an optimal solution

Probability of success:

$$\mathbb{P}[S] = \sum_{i=1}^{n} \mathbb{P}[S|1 \text{ is at pos i}]\mathbb{P}[1 \text{ is at pos i}]$$

$$= \frac{1}{n}\sum_{i=r}^{n} \mathbb{P}[S|1 \text{ is at pos i}]$$



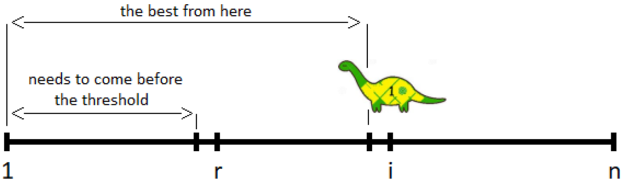
Secretary problem - an optimal solution

Probability of success:

$$\mathbb{P}[S] = \sum_{i=1}^{n} \mathbb{P}[S|1 \text{ is at pos i}]\mathbb{P}[1 \text{ is at pos i}]$$

$$= \frac{1}{n}\sum_{i=r}^{n} \mathbb{P}[S|1 \text{ is at pos i}]$$

$$= \frac{1}{n}\sum_{i=r}^{n} \frac{r-1}{i-1} = \frac{r-1}{n}\sum_{i=r}^{n}\frac{1}{i-1} \approx \frac{r}{n}\ln\frac{n}{r}$$



Secretary problem - an optimal solution

Probability of success: $\mathbb{P}[S] \approx \frac{r}{n} \ln \frac{n}{r}$.
The above function is maximised for $r = n/e$. Then $\mathbb{P}[S] \approx 1/e$.

Secretary problem - an optimal solution

Probability of success: $\mathbb{P}[S] \approx \frac{r}{n} \ln \frac{n}{r}$.
The above function is maximised for $r = n/e$. Then $\mathbb{P}[S] \approx 1/e$.

Asymptotically optimal solution:

- wait until a threshold $r_n \sim n/e$,
- afterwards accept the first candidate best so far,
- $\mathbb{P}[S] \xrightarrow{n\to\infty} 1/e \approx 0.368$.

Secretary problem - an optimal solution

Probability of success: $\mathbb{P}[S] \approx \frac{r}{n} \ln \frac{n}{r}$.
The above function is maximised for $r = n/e$. Then $\mathbb{P}[S] \approx 1/e$.

Asymptotically optimal solution:

- wait until a threshold $r_n \sim n/e$,
- afterwards accept the first candidate best so far,
- $\mathbb{P}[S] \xrightarrow{n\to\infty} 1/e \approx 0.368$.

Exact solution:

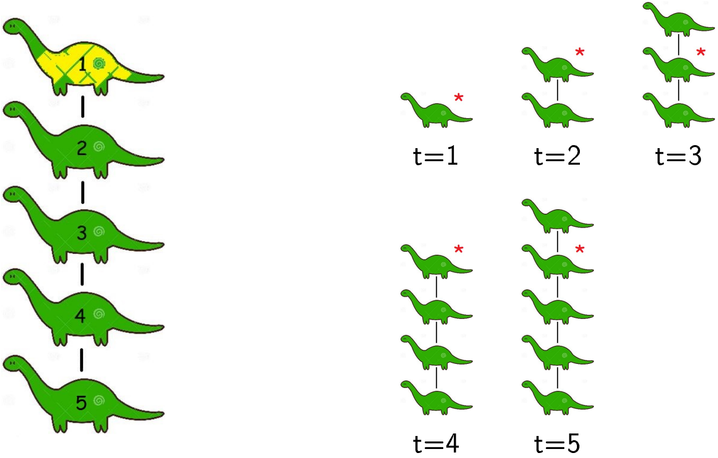
- wait until the first r such that $\frac{1}{r} + \frac{1}{r+1} + \ldots + \frac{1}{n-1} < 1$,
- at this time or later accept the first candidate best so far.

n	3	4	5	6	7	8	9	10	20	30	40	50	100
r	2	2	3	3	3	4	4	4	8	12	16	19	38
$\mathbb{P}[S]$	0.5	0.46	0.43	0.43	0.41	0.41	0.41	0.4	0.38	0.38	0.38	0.37	0.37

(Lindley, 1961)

Example revisited

Random permutation: $\sigma = (5, 3, 4, 1, 2)$ (you do not know it!)
For $n = 5$ the decision threshold is $r = 3$ $(1/3 + 1/4 \approx 0.58 < 1)$.



Countless generalizations...

- A postdoc problem (Rose 1982; Vanderbei, 2012).
  Maximise the probability of choosing **the second best**.

$$\mathbb{P}[S] \xrightarrow{n\to\infty} 1/4$$

Countless generalizations...

- A postdoc problem (Rose 1982; Vanderbei, 2012).
  Maximise the probability of choosing **the second best**.
  $$\mathbb{P}[S] \xrightarrow{n\to\infty} 1/4$$
- Minimise **the expected absolute rank** (Chow et al., 1964).
  $$\mathbb{E}[rank] \xrightarrow{n\to\infty} 3.8695....$$

Countless generalizations...

- A postdoc problem (Rose 1982; Vanderbei, 2012).
  Maximise the probability of choosing **the second best**.
  $$\mathbb{P}[S] \xrightarrow{n\to\infty} 1/4$$
- Minimise **the expected absolute rank** (Chow et al., 1964).
  $$\mathbb{E}[rank] \xrightarrow{n\to\infty} 3.8695....$$
- Success: accepting any of $\{1, 2, \ldots, k\}$ (Gusein-Zade, 1966).
  $$\text{for } k = 2 \quad \mathbb{P}[S] \xrightarrow{n\to\infty} 0.5736...$$

Countless generalizations...

- A postdoc problem (Rose 1982; Vanderbei, 2012).
  Maximise the probability of choosing **the second best**.
  $$\mathbb{P}[S] \xrightarrow{n\to\infty} 1/4$$

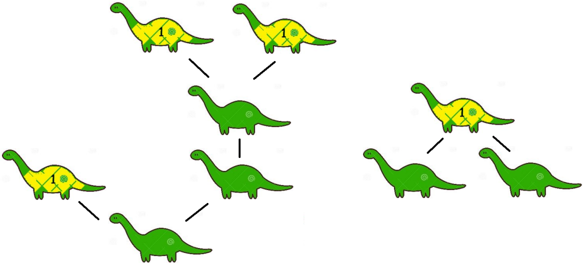
- Minimise **the expected absolute rank** (Chow et al., 1964).
  $$\mathbb{E}[rank] \xrightarrow{n\to\infty} 3.8695....$$

- Success: accepting any of $\{1, 2, \ldots, k\}$ (Gusein-Zade, 1966).
  $$\text{for } k = 2 \quad \mathbb{P}[S] \xrightarrow{n\to\infty} 0.5736...$$

- The possibility of **recall** and the possibility of **being refused** (Yang, 1974; Smith, 1975; Petrucelli, 1981)
  $$p_{being\_refused} = 1/2 \quad \mathbb{P}[S] \xrightarrow{n\to\infty} 1/4$$

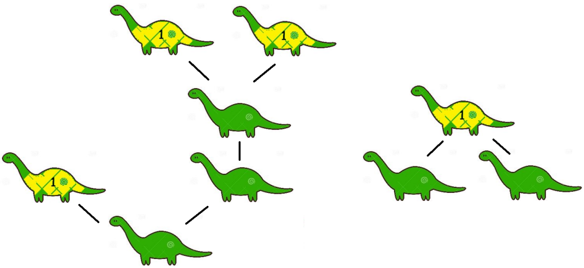
Countless generalizations...

- Partially ordered sets instead of a linear order (Stadje, 1980).



Countless generalizations...

- Partially ordered sets instead of a linear order (Stadje, 1980).



The selector knows the cardinality of the set but not its structure.
Success: stopping at any maximal element.

$$\mathbb{P}[S] > 1/e$$

(Freij, Wästlund, 2010)

Introduction to online algorithms in three steps

1. SECRETARY PROBLEM
2. SKI RENTAL PROBLEM - your homework!

Ski rental problem

- Rent per day: 1 \$.
- Cost of buying the equipment: k \$.
- You do not know how many days are you going to ski...

Should you buy? Should you rent? Should you rent and later buy?

Ski rental problem

- Rent per day: 1 \$.
- Cost of buying the equipment: k \$.
- You do not know how many days are you going to ski...

Should you buy? Should you rent? Should you rent and later buy?

OFF Optimal offline algorithm

j - number of skiing days

- if $j \leqslant k$ rent every day,
- if $j \geqslant k$ buy on the first day.

$$cost(OFF(j)) = \min\{j, k\}$$

Ski rental problem

- Rent per day: 1 \$.
- Cost of buying the equipment: k \$.
- You do not know how many days are you going to ski...

Should you buy? Should you rent? Should you rent and later buy?

OFF Optimal offline algorithm

j - number of skiing days

- if $j \leqslant k$ rent every day,
- if $j \geqslant k$ buy on the first day.

$$cost(OFF(j)) = \min\{j, k\}$$

Instances of our problem:
$\{1, 2, 3, \ldots\}$ - # of skiing days.

Strictly c-competitive online algorithm

For $c \geqslant 1$ ONLINE is strictly c-competitive if

$$\frac{cost(ONLINE(I))}{cost(OFF(I))} \leqslant c$$

for all instances I of the problem.

Ski rental problem

Examples

1. I buy on the first day.

$$cost(ONLINE(j)) = k \quad \text{for all} \quad j$$

$$\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))} = \frac{cost(ONLINE(1))}{cost(OFF(1))} = \frac{k}{1} = k$$

This algorithm is strictly k-competitive.

Ski rental problem

Examples

1. I buy on the first day.

$$cost(ONLINE(j)) = k \quad \text{for all} \quad j$$

$$\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))} = \frac{cost(ONLINE(1))}{cost(OFF(1))} = \frac{k}{1} = k$$

This algorithm is strictly k-competitive.

2. I always rent.

$$cost(ONLINE(j)) = j \quad \text{for all} \quad j$$

$$\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))} \overset{j \geqslant k}{=} \max_j \frac{j}{k} \xrightarrow{j \to \infty} \infty$$

This algorithm is not competitive at all...

Ski rental problem

Examples

1. I buy on the first day.

$$cost(ONLINE(j)) = k \quad \text{for all} \quad j$$

$$\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))} = \frac{cost(ONLINE(1))}{cost(OFF(1))} = \frac{k}{1} = k$$

   This algorithm is strictly k-competitive.

2. I always rent.

$$cost(ONLINE(j)) = j \quad \text{for all} \quad j$$

$$\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))} \overset{j \geqslant k}{=} \max_j \frac{j}{k} \xrightarrow{j \to \infty} \infty$$

   This algorithm is not competitive at all...

Let's find the best possible (in terms of strictly competitiveness) deterministic algorithm!

Ski rental problem, deterministic algorithm

Deterministic online algorithm is described just buy one number $t \geqslant 1$:

rent for $t - 1$ days and buy equipment on the t^{th} day

Ski rental problem, deterministic algorithm

Deterministic online algorithm is described just buy one number $t \geqslant 1$:

rent for $t - 1$ *days and buy equipment on the* t^{th} *day*

It's your turn! Find the optimal t!
Winter is coming...

Ski rental problem, deterministic algorithm

Deterministic online algorithm is described just buy one number $t \geqslant 1$:

rent for $t-1$ days and buy equipment on the t^{th} day

It's your turn! Find the optimal t!
Winter is coming...

Thank you!!! :)

Małgorzata Sulkowska 

- 2012-2014, **assistant** at the Wrocław University of Science and Technology, Poland
- 2013, **PhD**, *Stopping algorithms* under the supervision of prof. Michał Morayne Wrocław University of Science and Technology, Poland
- since 2014, **assistant professor** at the Wrocław University of Science and Technology, Poland
- 2015/2016, **postdoc** at the Federal University of Ceará, Brazil
- since 2020, **postdoc** at Université Côte d'Azur, Inria, CNRS, I3S, France

References

Y.S. Chow, S. Moriguti, H. Robbins, and S.M. Samuels.
Optimal selection based on relative rank (the secretary problem).
Israel J. Math., 2:81––90, 1964.

T.S. Ferguson.
Who solved the secretary problem?
Statist. Sci., 4(3):282–289, 08 1989.

P.R. Freeman.
The secretary problem and its extensions
Int. Stat. Rev., 51(2):189–206, 1983.

R. Freij and J. Wästlund.
Partially ordered secretaries.
Electron. Commun. Prob., 15:504–507, 2010.

S.M. Gusein-Zade.
The problem of choice and the optimal stopping rule for a sequence of independent trials.
Theory Probab. Appl+, 11(3):472––476, 1965.

D.V. Lindley.
Dynamic programming and decision theory.
Appl. Stat. - J. Roy. St. C, 10(1):39–51, 1961.

J.S. Rose.
A problem of optimal choice and assignment.
Operations Research, 30(1):172––181, 1982.

M. H. Smith.
A secretary problem with uncertain employment.
J. Appl. Prob., 12(3):620–624, 1975.

W. Stadje.
Efficient stopping of a random series of partially ordered points.
Multiple Criteria Decision Making Theory and Application. Lecture Notes in Economics and Mathematical Systems, 177:430–447, 1980.

R.J. Vanderbei.
The postdoc variant of the secretary problem, 2012.

M.C.K. Yang.
Recognizing the maximum of a random sequence based on relative rank with backward solicitation.
J. Appl. Prob., 11:504––512, 1974.

R.A. Baezayates, J.C. Culberson, and G.J.E. Rawlins.
Searching in the plane.
Inform. Comput., 106(2):234–252, 1993.

M.Y. Kao, J.H. Reif, and S.R. Tate.
Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem.
Inform. Comput., 131(1):63–79, 1996.

Sources of images

- https://www.scotthyoung.com/blog/2019/01/09/mih-decision-algorithm/
- https://commons.wikimedia.org/wiki/File:Random_car_parking_problem.svg
- https://pl.depositphotos.com/vector-images/car-broken-down.html
- https://tr.pinterest.com/pin/498703358732529489/
- https://www.pinterest.com/pin/600949762669473339/
- https:
  //pl.depositphotos.com/71415921/stock-illustration-i-small-lost-cow.html
- https://www.istockphoto.com/fr/search/2/image?mediatype=illustration&
  phrase=confused
- https://www.istockphoto.com/fr/search/2/image?mediatype=illustration&
  phrase=spiral
- https://www.istockphoto.com/fr/search/2/image?phrase=polish+flag

Ski rental problem, deterministic algorithm

Deterministic online algorithm is described just buy one number $t \geqslant 1$:

$$\textit{rent for } t - 1 \textit{ days and buy equipment on the } t^{th} \textit{ day}$$

Ski rental problem, deterministic algorithm

Deterministic online algorithm is described just buy one number $t \geqslant 1$:

rent for $t-1$ days and buy equipment on the t^{th} day

$$cost(ONLINE(j)) = \begin{cases} j, & j < t \\ t-1+k, & j \geqslant t \end{cases}$$

Ski rental problem, deterministic algorithm

Deterministic online algorithm is described just buy one number $t \geqslant 1$:

rent for $t-1$ days and buy equipment on the t^{th} day

$$cost(ONLINE(j)) = \begin{cases} j, & j < t \\ t-1+k, & j \geqslant t \end{cases}$$

Let's investigate $\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))}$ t by t:

- $t = 1$ then $c = k$ by choosing $j = 1$

Ski rental problem, deterministic algorithm

Deterministic online algorithm is described just buy one number $t \geqslant 1$:

rent for $t-1$ days and buy equipment on the t^{th} day

$$cost(ONLINE(j)) = \begin{cases} j, & j < t \\ t-1+k, & j \geqslant t \end{cases}$$

Let's investigate $\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))}$ t by t:

- $t = 1$ then $c = k$ by choosing $j = 1$
- $t = 2$ then $c = \frac{1+k}{2}$ by choosing $j = 2$

Ski rental problem, deterministic algorithm

Deterministic online algorithm is described just buy one number $t \geqslant 1$:

rent for $t-1$ days and buy equipment on the t^{th} day

$$cost(ONLINE(j)) = \begin{cases} j, & j < t \\ t - 1 + k, & j \geqslant t \end{cases}$$

Let's investigate $\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))}$ t by t:

- $t = 1$ then $c = k$ by choosing $j = 1$
- $t = 2$ then $c = \frac{1+k}{2}$ by choosing $j = 2$
- ...
- $t = k - 1$ then $c = \frac{k-2+k}{k-1}$ by choosing $j = k - 1$

Ski rental problem, deterministic algorithm

Deterministic online algorithm is described just buy one number $t \geqslant 1$:

rent for $t-1$ days and buy equipment on the t^{th} day

$$cost(ONLINE(j)) = \begin{cases} j, & j < t \\ t-1+k, & j \geqslant t \end{cases}$$

Let's investigate $\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))}$ t by t:

- $t = 1$ then $c = k$ by choosing $j = 1$
- $t = 2$ then $c = \frac{1+k}{2}$ by choosing $j = 2$
- ...
- $t = k-1$ then $c = \frac{k-2+k}{k-1}$ by choosing $j = k-1$
- $t = k$ then $c = \frac{k-1+k}{k}$ by choosing any $j \geqslant k$

Ski rental problem, deterministic algorithm

Deterministic online algorithm is described just buy one number $t \geqslant 1$:

rent for $t-1$ *days and buy equipment on the* t^{th} *day*

$$cost(ONLINE(j)) = \begin{cases} j, & j < t \\ t-1+k, & j \geqslant t \end{cases}$$

Let's investigate $\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))}$ t by t:

- $t = 1$ then $c = k$ by choosing $j = 1$
- $t = 2$ then $c = \frac{1+k}{2}$ by choosing $j = 2$
- ...
- $t = k-1$ then $c = \frac{k-2+k}{k-1}$ by choosing $j = k-1$
- $t = k$ then $c = \frac{k-1+k}{k}$ by choosing any $j \geqslant k$
- $t = k+1$ then $c = \frac{k+k}{k}$ by choosing any $j \geqslant k+1$
- ...

Ski rental problem, deterministic algorithm

Let's investigate $\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))}$ t by t:

- $t = 1$ then $c = k$ by choosing $j = 1$
- $t = 2$ then $c = \frac{1+k}{2}$ by choosing $j = 2$
- ...
- $t = k - 1$ then $c = \frac{k-2+k}{k-1}$ by choosing $j = k - 1$
- $t = k$ then $c = \frac{k-1+k}{k}$ by choosing any $j \geqslant k$
- $t = k + 1$ then $c = \frac{k+k}{k}$ by choosing any $j \geqslant k$
- ...

We get $$c = \begin{cases} \frac{t-1+k}{t}, & t \leqslant k \\ \frac{t-1+k}{k}, & t \geqslant k \end{cases} = \begin{cases} 1 + \frac{k-1}{t}, & t \leqslant k \\ 1 + \frac{t-1}{k}, & t \geqslant k \end{cases}.$$

Ski rental problem, deterministic algorithm

Let's investigate $\max_j \frac{cost(ONLINE(j))}{cost(OFF(j))}$ t by t:

- $t = 1$ then $c = k$ by choosing $j = 1$
- $t = 2$ then $c = \frac{1+k}{2}$ by choosing $j = 2$
- ...
- $t = k-1$ then $c = \frac{k-2+k}{k-1}$ by choosing $j = k-1$
- $t = k$ then $c = \frac{k-1+k}{k}$ by choosing any $j \geqslant k$
- $t = k+1$ then $c = \frac{k+k}{k}$ by choosing any $j \geqslant k$
- ...

We get $$c = \begin{cases} \frac{t-1+k}{t}, & t \leqslant k \\ \frac{t-1+k}{k}, & t \geqslant k \end{cases} = \begin{cases} 1 + \frac{k-1}{t}, & t \leqslant k \\ 1 + \frac{t-1}{k}, & t \geqslant k \end{cases}.$$

Choosing $t = k$ we get strictly $(2 - 1/k)$-competitive algorithm.

Ski rental problem

- Choosing $t = k$ we get strictly $(2 - 1/k)$-competitive algorithm:

  rent for $k - 1$ days and buy equipment on the k^{th} day.

  This is the best deterministic algorithm.

Ski rental problem

- Choosing $t = k$ we get strictly $(2 - 1/k)$-competitive algorithm:

  rent for $k - 1$ days and buy equipment on the k^{th} day.

  This is the best deterministic algorithm.

Can we do better?

Ski rental problem

- Choosing $t = k$ we get strictly $(2 - 1/k)$-competitive algorithm:

  rent for $k - 1$ days and buy equipment on the k^{th} day.

  This is the best deterministic algorithm.

Can we do better?

Yes! Randomization helps!

Ski rental problem, randomized algorithm

- Set of deterministic strategies: $\{S_1, S_2, \ldots, S_k\}$
  S_i : *rent for* $i - 1$ *days and buy equipment on the* i^{th} *day.*
- Probability distribution: $\{p_1, p_2, \ldots, p_k\}$
- Randomized algorithm RAND chooses strategy S_i with probability p_i.

Ski rental problem, randomized algorithm

- Set of deterministic strategies: $\{S_1, S_2, \ldots, S_k\}$
  S_i : *rent for* $i-1$ *days and buy equipment on the* i^{th} *day.*
- Probability distribution: $\{p_1, p_2, \ldots, p_k\}$
- Randomized algorithm RAND chooses strategy S_i with probability p_i.

Strictly c-competitive online randomized algorithm

For $c \geqslant 1$ RAND is strictly c-competitive if

$$\frac{\mathbb{E}[cost(RAND(I))]}{cost(OFF(I))} \leqslant c$$

for all instances I of the problem.

Ski rental problem, randomized algorithm

- Set of deterministic strategies: $\{S_1, S_2, \ldots, S_k\}$
  S_i : *rent for* $i-1$ *days and buy equipment on the* i^{th} *day.*
- Probability distribution: $\{p_1, p_2, \ldots, p_k\}$
- Randomized algorithm RAND chooses strategy S_i with probability p_i.

Strictly c-competitive online randomized algorithm

For $c \geqslant 1$ RAND is strictly c-competitive if

$$\frac{\mathbb{E}[cost(RAND(I))]}{cost(OFF(I))} \leqslant c$$

for all instances I of the problem.

- The best probability distribution

$$p_i = \gamma \delta^{i-1}$$

$$\delta = \frac{k}{k-1} \qquad \gamma = \frac{\delta - 1}{\delta^k - 1}$$

Ski rental problem, randomized algorithm

- Set of deterministic strategies: $\{S_1, S_2, \dots, S_k\}$
  S_i : *rent for* $i-1$ *days and buy equipment on the* i^{th} *day.*
- Probability distribution: $\{p_1, p_2, \dots, p_k\}$
- Randomized algorithm RAND chooses strategy S_i with probability p_i.

Strictly c-competitive online randomized algorithm

For $c \geqslant 1$ RAND is strictly c-competitive if

$$\frac{\mathbb{E}[cost(RAND(I))]}{cost(OFF(I))} \leqslant c$$

for all instances I of the problem.

- The best probability distribution

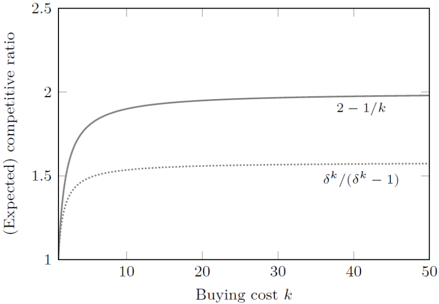
$$p_i = \gamma\delta^{i-1}$$

$$\delta = \frac{k}{k-1} \qquad \gamma = \frac{\delta - 1}{\delta^k - 1}$$

- Then RAND is strictly $\frac{\delta^k}{\delta^k - 1}$-competitive

$$\frac{\delta^k}{\delta^k - 1} \xrightarrow{k\to\infty} \frac{e}{e-1} \approx 1.582$$

Ski rental problem, comparison of algorithms



- The best deterministic online algorithm is strictly $(2 - \frac{1}{k})$-competitive.

$$2 - 1/k \xrightarrow{k \to \infty} 2$$

- The best randomized online algorithm is strictly $\frac{\delta^k}{\delta^k - 1}$-competitive, where $\delta = \frac{k}{k-1}$

$$\frac{\delta^k}{\delta^k - 1} \xrightarrow{k \to \infty} \frac{e}{e-1} \approx 1.582$$

Thank you again!!! :)