Search under Uncertainty & Doubling Technique

Shahin Kamali

Lecture 2 - Sep. 10, 2019

University of Manitoba
While waiting, in case you have your laptop/cell-phone with you:

- go to https://www.iclicker.com/students and register/log-in as a student (or install iClicker Reef on your phone).
- find “Online Algorithms” course at “University of Manitoba”, and add it to your courses!
- join the class at the beginning of the class
- respond to the following question in the quiz.

**Question:** indicate which animal below is “cuter”?

(a)  
(b)  
(c)  
(d)
Review
Offline vs. Online Algorithms

- Traditional algorithms are ‘offline’ in the sense that they have the whole input in their hand.

- Online algorithms, in contrast, do not have/need the whole input in order to solve a problem
  - The online algorithms often take irrevocable decisions to process the input.
Online algorithms are

- Practical
- Diverse
- Fun (really!)
Online algorithms are

- Practical
- Diverse
- Fun (really!)

Let’s ‘play’ with online algorithms and enjoy
Logistics

Logistics

Logistics

- Office hours: Tuesdays, 2:00pm - 3:00pm Thursdays, 11:30am-12:30pm or by appointment (in E2 586)
Ski-Rental Problem

Ski-rental problem

- Assume you want to go skiing for \( x \) number of days
  - In the online setting, the value of \( x \) is unknown!
Ski-Rental Problem

Ski-rental problem

- Assume you want to go skiing for $x$ number of days
  - In the online setting, the value of $x$ is unknown!
- You can buy the equipment for a one-time cost of $b$ or rent each day for a cost of 1 per day
Ski-Rental Problem

Ski-rental problem

- Assume you want to go skiing for \( x \) number of days
  - In the online setting, the value of \( x \) is unknown!
- You can buy the equipment for a one-time cost of \( b \) or rent each day for a cost of 1 per day
- If we know \( x \), what is the best solution?
Ski-Rental Problem

Assume you want to go skiing for \( x \) number of days

- In the online setting, the value of \( x \) is unknown!

You can buy the equipment for a one-time cost of \( b \) or rent each day for a cost of 1 per day

If we know \( x \), what is the best solution?

- Buy at the beginning if \( x \geq b \), otherwise, rent every day
Ski-Rental Problem

Ski-rental problem

- Assume you want to go skiing for $x$ number of days
  - In the online setting, the value of $x$ is unknown!
- You can buy the equipment for a one-time cost of $b$ or rent each day for a cost of 1 per day
- If we know $x$, what is the best solution?
  - Buy at the beginning if $x \geq b$, otherwise, rent every day
- What is the competitive ratio of an algorithm that buys at day 1?
Ski-Rental Problem

Assume you want to go skiing for $x$ number of days

- In the online setting, the value of $x$ is unknown!

You can buy the equipment for a one-time cost of $b$ or rent each day for a cost of 1 per day

If we know $x$, what is the best solution?

- Buy at the beginning if $x \geq b$, otherwise, rent every day

What is the competitive ratio of an algorithm that buys at day 1?

- In the worst case, you go skiing once; so $\frac{b}{1} = b$ (not good)
Ski-Rental Problem

### Ski-rental problem

- Assume you want to go skiing for $x$ number of days
  - In the online setting, the value of $x$ is unknown!
- You can buy the equipment for a one-time cost of $b$ or rent each day for a cost of 1 per day
- If we know $x$, what is the best solution?
  - Buy at the beginning if $x \geq b$, otherwise, rent every day
- What is the competitive ratio of an algorithm that buys at day 1?
  - In the worst case, you go skiing once; so $\frac{b}{1} = b$ (not good)
- What is the competitive ratio of an algorithm that always rent?
  - In the worst-case, we go skiing $n$ days for large $n$
  - The competitive ratio is $\frac{n}{b}$, which can be arbitrary large (very bad).
Online strategy **break-even**: rent for the first $b - 1$ days and buy in the next day.
Online strategy **break-even**: rent for the first $b - 1$ days and buy in the next day.

What is the competitive ratio of Break-even algorithm?
Online strategy **break-even**: rent for the first $b - 1$ days and buy in the next day.

What is the competitive ratio of Break-even algorithm?

It is $\frac{(b-1)+b}{b} \approx 2$

**Theorem**

*Competitive ratio is roughly 2, and it is the best for any deterministic online algorithm.*
Cow-Path Problem
Problem Definition

- A cow faces a fence, infinite in both directions.
- She wants to find a hole in order to get to the green pasture on the other side.
- The cows' online strategy specifies the path traveled in search of the hole.
- The goal is to minimize the distance traveled.
Let \( u \) an integer indicating the distance between the initial location of the cow and the location of the hole. \( u \) is unknown to the cow!
Let $u$ an integer indicating the distance between the initial location of the cow and the location of the hole.

- $u$ is unknown to the cow!

An optimal offline algorithm $Opt$ (i.e., a cow which knows the location of the hole), incurs a cost of $u$. 
Cow-Path Problem

Smart-Cow Algorithm (SCA)

- Gradually extend the explored interval of the fence
- Alternate between left and right!
  - Go right for distance $d_0$
  - Go back to the origin, left for distance $d_1$
  - Go back to the origin, right for distance $d_2$
  - Continue accordingly for $d_3, \ldots, d_k$ until the hole is found.
Recall that the competitive ratio of an online algorithm is the maximum ratio between the cost of that algorithm and an optimal offline $\text{OPT}$ algorithm $\text{OPT}$.

- The cost of $\text{OPT}$ is $u$.
- The cost of SCA is $2d_0 + 2d_1 + \ldots + 2d_{k-2} + 2d_{k-1} + u$
  - $d_{k-2} < u \leq d_k$.
Cow-Path Problem

Competitive Ratio of SCA (cntd.)

- The competitive ratio would be

\[
\frac{2d_0 + 2d_1 + \ldots + 2d_{k-2} + 2d_{k-1} + u}{u} = 1 + 2 \frac{d_0 + d_1 + \ldots + d_{k-1}}{u}
\]

- What is the value of \( u \) in the worst case?

  - If you are an adversary and want to fail the algorithm, where you place the hole?
The competitive ratio would be
\[
\frac{2d_0 + 2d_1 + \ldots + 2d_{k-2} + 2d_{k-1} + u}{u} = 1 + 2 \frac{d_0 + d_1 + \ldots + d_{k-1}}{d_{k-2} + \epsilon}
\]

In the worst case, \( u = d_{k-2} + \epsilon \).

Just a bit more than the previous probe!

So, the competitive ratio of a Smart-Cow algorithm is
\[
1 + 2 \frac{d_0 + d_1 + \ldots + d_{k-1}}{d_{k-2} + \epsilon}
\]
The Doubling Technique

- Assume \( d_i = 2^i \), i.e., first go one unit to the right, go back to the origin, go two units to the left, back to origin, four units to the right, etc.

- We will have
  \[
  d_0 + d_1 + \ldots + d_{k-1} = 1 + 2 + 4 + \ldots + 2^{k-1} = 2^k - 1 = 4 \cdot 2^{k-2}.
  \]

- The competitive ratio would be
  \[
  1 + 2 \frac{d_0 + d_1 + \ldots + d_{k-1}}{d_{k-2} + \epsilon} = 1 + 2 \frac{4 \cdot 2^{k-2}}{2^{k-2} + \epsilon} \approx 9
  \]
The smart-cow algorithm with steps that double (i.e., $d_i = 2^i$) has a competitive ratio of at most 9.
Theorem

*The smart-cow algorithm with steps that double (i.e., \( d_i = 2^i \)) has a competitive ratio of at most 9.*

- It turns out that no *deterministic* algorithm can achieve a ratio better than 9.
  - The proof is a bit involved and we skip it here.
Theorem

The smart-cow algorithm with steps that double (i.e., $d_i = 2^i$) has a competitive ratio of at most 9.

- It turns out that no deterministic algorithm can achieve a ratio better than 9.
  - The proof is a bit involved and we skip it here.
- So, the doubling technique results an optimal algorithm in this case.
We assumed the value of \( u \) is unknown to the algorithm.

**Question:** what competitive an “almost-online” algorithm can achieve when the value of \( u \) is known?

- The algorithm knows \( u \) but does not know the side (left or right) where the target is located.

(a) 1   (b) 2   (c) 3   (d) 4   (f) 9   (g) 12
A cow can be a robot (or the other way around)!

In practice, robots often do not have full information about their environment.

Cow-path problem and its variant are a way to model many types of search problems.
Path-cow problem is an online search problem on a path.

Consider a star, where \( w \) paths have one common endpoint.

Assume a robot is initially located at the common point, and needs to find a target located in an unknown position.

What is a good algorithm?
The best strategy is to have
\[ d_i = \left( \frac{w}{w - 1} \right)^i. \]
- For \( w = 2 \), it requires doubling.
- For \( w = 3 \), we jump by a factor of \( 3/2 \), and so on.

The competitive ratio will be at most
\[ 1 + \frac{2}{\sqrt{e}} \left( \frac{w}{w - 1} \right)^i \approx 1 + \frac{2}{\sqrt{e}} \left( \frac{1}{1} \right)^i \]
when \( w \) is sufficiently large.

Note that doubling is not optimal here. But it is still competitive, i.e., it has a constant competitive ratio.

\( e \approx 2.71 \) is the Euler’s constant.
The best strategy is to have
\[ d_i = \left( \frac{w}{w - 1} \right)^i. \]
- For \( w = 2 \), it requires doubling.
- For \( w = 3 \), we jump by a factor of \( 3/2 \), and so on.

The competitive ratio will be at most
\[ 1 + 2 \frac{w}{(w-1)^{w-1}} \approx 1 + 2e(w - 1) \] (when \( w \) is sufficiently large).
- \( e \approx 2.71 \) is the Euler’s constant
Cow-Path Problem

Variants of Search Problems

- The best strategy is to have
  \[ d_i = \left( \frac{w}{w - 1} \right)^i. \]

  - For \( w = 2 \), it requires doubling.
  - For \( w = 3 \), we jump by a factor of \( \frac{3}{2} \), and so on.

- The competitive ratio will be at most
  \[ 1 + 2\left( \frac{w^w}{(w-1)^{w-1}} \right) \approx 1 + 2e(w - 1) \] (when \( w \) is sufficiently large).

  - \( e \approx 2.71 \) is the Euler’s constant

- Note that doubling is not optimal here.

  - But it is still \textbf{competitive}, i.e., it has a constant competitive ratio.
Randomized Online Algorithms
Randomization often helps online algorithms to achieve better competitive ratios.

In competitive analysis of online algorithms, we consider worst-case inputs generated by an adversary.
Randomized Online Algorithms

Randomization often helps online algorithms to achieve better competitive ratios.

In competitive analysis of online algorithms, we consider worst-case inputs generated by an adversary.

For randomized algorithms, we compare online algorithms against an oblivious adversary which is unaware of random choices made by the algorithms.

- The adversary knows the code of the algorithm but does not know the run-time random bits used by the algorithm.
There are two mirroring algorithms for the deterministic algorithm.

The worst-case position of the hole is not the same for these two algorithms.

Randomized-Smart-Cow: chose one of the mirroring algorithm uniformly at random.
In order to find the expected cost of the algorithm, we find the summation of costs of the two mirroring algorithms.

w.l.o.g. assume the hole is located on the right

\[ \text{The expected cost of the algorithm is } 2^k - 1 + 2^k - 1 - 1 + u. \]

The competitive ratio is at most

\[ \frac{2^k + 2^k - 1 + u - 2u}{2^k} = 1 + \frac{2^k - 1 + u}{2^k}. \]

In the worst case, \( u = 2^k - 2 + \epsilon \) which gives a competitive ratio of at most 7.
In order to find the expected cost of the algorithm, we find the summation of costs of the two mirroring algorithms.

- w.l.o.g. assume the hole is located on the right
- The total sum on the left of the origin is $2(1 + 2 + \ldots + 2^{k-1}) = 2(2^k - 1)$.

The expected cost of the algorithm is $2^k - 1 + 2^k - 1 - 1 + \epsilon$. 

The competitive ratio is at most $2^k + 2^k - 1 + \epsilon - 2^k = 1 + 2^k + 2^k - 1 - 2^k$. 

In the worst case, $\epsilon = 2^k - 2 + \epsilon$ which gives a competitive ratio of at most 7.
In order to find the expected cost of the algorithm, we find the summation of costs of the two mirroring algorithms.

- w.l.o.g. assume the hole is located on the right
- The total sum on the left of the origin is
  \[ 2(1 + 2 + \ldots + 2^{k-1}) = 2(2^k - 1). \]
- The total sum on the right of the origin is
  \[ 2(1 + 2 + \ldots + 2^{k-2}) + 2u = 2(2^{k-1} - 1 + u). \]
In order to find the expected cost of the algorithm, we find the summation of costs of the two mirroring algorithms.

- w.l.o.g. assume the hole is located on the right
- The total sum on the left of the origin is
  \[2(1 + 2 + \ldots + 2^{k-1}) = 2(2^k - 1).\]
- The total sum on the right of the origin is
  \[2(1 + 2 + \ldots + 2^{k-2}) + 2u = 2(2^{k-1} - 1 + u).\]
- The expected cost of the algorithm is \(2^k - 1 + 2^{k-1} - 1 + u\).
In order to find the expected cost of the algorithm, we find the summation of costs of the two mirroring algorithms.

- w.l.o.g. assume the hole is located on the right
- The total sum on the left of the origin is $2(1 + 2 + \ldots + 2^{k-1}) = 2(2^k - 1)$.
- The total sum on the right of the origin is $2(1 + 2 + \ldots + 2^{k-2}) + 2u = 2(2^{k-1} - 1 + u)$.
- The expected cost of the algorithm is $2^k - 1 + 2^{k-1} - 1 + u$.

The competitive ratio is at most $\frac{2^k + 2^{k-1} + u - 2}{u} = 1 + \frac{2^k + 2^{k-1} - 2}{u}$.
In order to find the expected cost of the algorithm, we find the summation of costs of the two mirroring algorithms.

w.l.o.g. assume the hole is located on the right
The total sum on the left of the origin is
\[ 2(1 + 2 + \ldots + 2^{k-1}) = 2(2^k - 1). \]
The total sum on the right of the origin is
\[ 2(1 + 2 + \ldots + 2^{k-2}) + 2u = 2(2^{k-1} - 1 + u). \]
The expected cost of the algorithm is \( 2^k - 1 + 2^{k-1} - 1 + u \).

The competitive ratio is at most \( \frac{2^k+2^{k-1}+u-2}{u} = 1 + \frac{2^k+2^{k-1}-2}{u} \).
In the worst case, \( u = 2^{k-2} + \epsilon \) which gives a competitive ratio of at most 7.
With only one random bit, we achieve a competitive ratio of 7, which is better than any deterministic algorithm can achieve.
With only one random bit, we achieve a competitive ratio of 7, which is better than any deterministic algorithm can achieve.

There is a more complicated randomized algorithm achieves a competitive ratio of 4.591.

- Smart-Cow with different jumps.
Online Algorithms with Advice
Online Algorithms with Advice

Advice Complexity of Online Algorithms

Assume an online algorithm receives some **bits of advice** from a benevolent oracle.

The advice can be anything that can help the algorithm its competitive ratio

What is a good advice for cow-path problem?
How many bits of advice are sufficient to achieve an optimal solution

What is the best competitive ratio that one can achieve with $c$ bits of advice?
How many bits are sufficient to achieve an optimal solution?
How many bits are sufficient to achieve an optimal solution?

- With $\Theta(\lceil \log w \rceil)$ bits, indicate the row at which the target is located.
- Indeed, at least $\lceil \log w \rceil$ bits are also required to be optimal!
What can we do with $c < \log w$ bits of advice?

- Recall that the competitive ratio of smart-cow with $d_i = (w/(w - 1))^i$ is roughly $1 + 2e(w - 1)$.

Assume we can have 1 bit of advice!

- Partition the $w$ paths into two groups of size $w/2$.
- The advice to specifies in which group the target is located.
- The competitive ratio becomes $1 + 2e(w/2 - 1)$.

With $c$ bits, the competitive ratio will be $1 + 2e\left(\frac{w}{2^c} - 1\right)$.
Variants of Online Search Problems

Potential Topics for Final Projects
Variants of Online Search Problems

Online Search on Trees and Tree-like Graphs

- Assume a robot needs to search on a tree instead of a path or a star.
- What is the competitive ratio of smart-cow with doubling jumps?
  - The tree might or might not be regular, i.e., all internal nodes have an equal $k$ neighbors.
- What about graphs which are almost trees?
  - e.g., Cactus graphs?
- How randomization and advice can help tackling these problems?
Two-dimensional Cow-Path Problem

Assume the cow is in a 2d plane and wants to find a hole in an (unseen) fence.
Two-dimensional Cow-Path Problem

- Assume the cow is in a 2d plane and wants to find a hole in an (unseen) fence.
- Maybe we should make jumps in different directions?
- There are many variants of this problem!
Online Search with Moving Objects

- Assume the ‘cow’ is a cat and the ‘hole’ is a mouse.
  - Cat moves faster than the mouse!
  - What is the strategy for the cat to catch the mouse while moving a minimum distance?

- What if they are located on a tree or a cactus?
- What if they are limited to a fence?
- How advice/randomization helps?
Conclusions
Online Search problems have many practical applications.
Concluding Remarks

- Online Search problems have many practical applications.
- They can be modeled with variants of Cow-path problem.
Online Search problems have many practical applications.

They can be modeled with variants of Cow-path problem.

Doubling technique and its variants often result in competitive algorithms.
Online Search problems have many practical applications.

They can be modeled with variants of Cow-path problem.

Doubling technique and its variants often result in competitive algorithms.

Randomization and advice are tools that can help improve competitive ratio of online algorithms.