Doubling Technique (cntd.): Online Bidding and Clustering

Shahin Kamali

Lecture 3 - Sep. 14, 2017

University of Manitoba
Review & Plan
Doubling Technique for Online Algorithms

- When you are searching for an unknown value/position, you can use doubling (or jumping by other factors) to guess the distance/cost that should be taken in the next step.

- This often results in competitive algorithms.
Doubling Technique for Online Algorithms

- When you are searching for an unknown value/position, you can use doubling (or jumping by other factors) to guess the distance/cost that should be taken in the next step.
  - This often results in competitive algorithms.

- In path cow problem, we double the distance moved in each step
  - Similar technique can be used to achieve competitive algorithms for many search problems.
Today’s objectives

- Online Bidding Problem
  - Upper and lower bound for competitive ratio of deterministic algorithms
  - An optimal randomized algorithm

- Online Clustering Problem
Online Bidding Problem
We face an unknown target $u$

A player (online algorithm) submits a sequence $d_0, \ldots, d_k$ of bids until one is greater than or equal to $u$.

- we have $d_0 < d_1 < \ldots < d_{k-1} < u \leq d_k$. 

The competitive ratio of the algorithm is ?  

Magical betting formula

$\frac{4}{15}$
We face an unknown target $u$.

A player (online algorithm) submits a sequence $d_0, \ldots, d_k$ of bids until one is greater than or equal to $u$.

- we have $d_0 < d_1 < \ldots < d_{k-1} < u \leq d_k$.

The competitive ratio of the algorithm is?
Online Bidding Problem

Problem Definition

- We face an unknown target $u$
- A player (online algorithm) submits a sequence $d_0, \ldots, d_k$ of bids until one is greater than or equal to $u$.
  - we have $d_0 < d_1 < \ldots < d_{k-1} < u \leq d_k$.
- The competitive ratio of the algorithm is?

Magical betting formula

$$c.r. = \max_{u,k} \left\{ \frac{d_0 + d_1 + \ldots + d_k}{u} \right\}$$
Online Bidding Problem

Doubling algorithm

- Begin with a bid of cost $d_0 = 1$ and ‘jump’ by a factor of 2 for each subsequent bid
  - we have $d_0 = 1$, $d_1 = 2$, ..., $d_i = 2^i$.
- What is the competitive ratio?
Assume we stop after guessing \( k \) items.

The cost of the algorithm is \( 1 + 2 + \ldots + 2^k = 2^{k+1} - 1 \).

The cost of \( \text{OPT} \) is \( u \) where \( 2^{k-1} < u \leq 2^k \).

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

- If you are adversary, where do you select \( u \) to be?
- The worse case is when \( u = 2^{k-1} + \epsilon \).

The competitive ratio is \( \frac{1+2+\ldots+2^k}{u} = \frac{2^{k+1} - 1}{2^{k-1} + \epsilon} \approx 4 \).
Can we do better?

Theorem

*The doubling algorithm has a competitive ratio of 4.*

- Is there any *deterministic* algorithm with a better competitive ratio?
Online Bidding Problem

Can we do better?

**Theorem**

*The doubling algorithm has a competitive ratio of 4.*

- Is there any *deterministic* algorithm with a better competitive ratio?
  - The answer is **No**

**Theorem**

*There is no deterministic algorithm for online bidding with a c.r. better than 4.*
Assume there is an algorithm with a competitive ratio $\alpha < 4$.

- Let $s_i = d_0 + d_1 + \ldots + d_i$ (the cost after $i$ steps).
- Let $y_i = \frac{s_i}{s_{i-1}}$, (e.g., for doubling algorithm $y_i$ approaches to 2 for large values of $i$.)

Since c.r. is at most $\alpha$, we have $\frac{s_{n+1}}{d_n} \leq \alpha$ for large values of $n$.

\[
\begin{align*}
    s_{n+1} &\leq \alpha d_n \\ 
    \frac{s_{n+1}}{s_n} &\leq \alpha \\ 
    \frac{d_n}{s_n} &\leq \alpha \\ 
    y_n &\leq \frac{s_n - s_{n-1}}{s_n} \\ 
    y_n &\leq \frac{s_n}{s_{n-1}} \\ 
    y_n &\leq \frac{s_{n+1}}{d_n} \cdot \alpha \\ 
    y_n &\leq \frac{1}{y_{n-1}} - 1
\end{align*}
\]

For any value of $x$, we have $(x - 2)^2 = x^2 - 4x + 4 \geq 0$, i.e., $1 - 1/x \leq x/4$. So, plugging $x = y_{n-1}$ we get $y_n \leq y_{n-1} \cdot \alpha/4$.

Since $\alpha < 4$, in each step, the value of $y_n$ is decreased by a fraction.

After a sufficiently large number of steps, we will have $y_n < 1$ which implies $\frac{s_n}{s_{n-1}} < 1$, i.e., $s_n < s_{n-1}$.

It contradicts the definition of $s_n$, so the initial assumption of $\alpha < 4$ does not hold!
The doubling algorithm has a c.r. of 4 for the online bidding problem.

No deterministic online algorithm can do better!
Theorem

The doubling algorithm has a c.r. of 4 for the online bidding problem.

No deterministic online algorithm can do better!

• What about a randomized algorithm?
Online Bidding Problem

Randomized Online Bidding

- The algorithm selects $X$ as a random number in the range $U[0, 1)$.
- The bids of the algorithm are
  $d_0 = e^X, d_1 = e^{X+1}, d_2 = e^{X+3}, \ldots, d_{k-1} = e^{X+k-1}, d_k = e^{X+k}$.
- What is the expected value for $J = d_k/u$?
  - Recall the worst-case value of $d_k/u$ for doubling algorithm was 2.
Online Bidding Problem

Randomized Online Bidding

• The algorithm selects $X$ as a random number in the range $U[0, 1)$.

• The bids of the algorithm are
  
  $d_0 = e^X$, $d_1 = e^{X+1}$, $d_2 = e^{X+3}$, $\ldots$, $d_{k-1} = e^{X+k-1}$, $d_k = e^{X+k}$.

• What is the expected value for $J = d_k/u$?

  • Recall the worst-case value of $d_k/u$ for doubling algorithm was 2.

  • Assume $u = e^p$, so we have: $u = e^p \leq e^{X+k} < e^{p+1}$

    • So, $p \leq X + k < p + 1$.

    • $X + k - p$ is a uniformly distributed random variable in $U[0, 1)$.
Online Bidding Problem

Randomized Online Bidding

- The algorithm selects $X$ as a random number in the range $U[0, 1)$.
- The bids of the algorithm are $d_0 = e^X$, $d_1 = e^{X+1}$, $d_2 = e^{X+3}$, ..., $d_{k-1} = e^{X+k-1}$, $d_k = e^{X+k}$.
- What is the expected value for $J = d_k/u$?
  - Recall the worst-case value of $d_k/u$ for doubling algorithm was 2.
- Assume $u = e^p$, so we have: $u = e^p \leq e^{X+k} < e^{p+1}$
  - So, $p \leq X + k < p + 1$.
  - $X + k - p$ is a uniformly distributed random variable in $U[0, 1)$.
- We have $J = d_k/u = e^{x+k}/e^p = e^{x+k-p}$, i.e., $J$ is a random variable distributed as $e^Y$ where $Y$ is uniform in $[0, 1)$.
- The expected value of $J$ is $\int_0^1 e^Y dY = e - 1 \approx 1.71$.
  - Note the improvement over doubling algorithm!
The bids of the algorithm are
\[d_0 = e^X, \ d_1 = e^{X+1}, \ldots, \ d_{k-1} = e^{X+k-1}, \ d_k = e^{X+k} \].

The expected value of \( J = d_k/u \) is \( e - 1 \).
Online Bidding Problem

Randomized Online Bidding (cntd.)

- The bids of the algorithm are
  \[ d_0 = e^X, \quad d_1 = e^{X+1}, \ldots, \quad d_{k-1} = e^{X+k-1}, \quad d_k = e^{X+k}. \]

- The expected value of \( J = d_k/u \) is \( e - 1 \).

- The expected cost of the algorithm would be
  \[
  e^X(1 + e + e^2 + \ldots + e^k) = e^X e^k (1 + 1/e + \ldots + 1/e^k)
  = e^{X+k} e/(e - 1) = d_k e/(e - 1)
  \]

- The competitive ratio would be
  \[
  \frac{E[\text{cost(alg)}]}{\text{cost(opt)}} = \frac{E[d_k e(e - 1)]}{u} = E\left[ \frac{d_k}{u} \right] e/(e-1) = (e-1) \cdot e/(e-1) = e
  \]
There is a randomized algorithm for online bidding which has c.r. of $e$.

Indeed, it is the best that a randomized algorithm can achieve.
Online Clustering Problem
Online Clustering Problem

Problem Definition

- Partition a set of points in the plane into $k$ clusters

$k=3$
Online Clustering Problem

Problem Definition

- Partition a set of points in the plane into \( k \) clusters
- The **diameter** of a cluster is the maximum distance between any two points in the cluster

\[ k = 3 \]
Online Clustering Problem

Problem Definition

- Partition a set of points in the plane into $k$ clusters
- The **diameter** of a cluster is the maximum distance between any two points in the cluster
- The objective is to achieve a clustering with minimum diameter.

$k=3$
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to merge clusters but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner
- At each time, we should have a partitioning of the appeared nodes into \( k \) clusters.
- We are allowed to **merge clusters** but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to merge clusters but we cannot break them!
Online Clustering

- The set of points appear in an online manner
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to merge clusters but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to merge clusters but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to merge clusters but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to merge clusters but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to merge clusters but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into \( k \) clusters.
- We are allowed to **merge clusters** but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to merge clusters but we cannot break them!

$k = 3$
Online Clustering

- The set of points appear in an online manner
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to **merge clusters** but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to **merge clusters** but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to **merge clusters** but we cannot break them!

$k = 3$
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into \( k \) clusters.
- We are allowed to merge clusters but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into \( k \) clusters.
- We are allowed to merge clusters but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to **merge clusters** but we cannot break them!

$k = 3$
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to merge clusters but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to **merge clusters** but we cannot break them!

$k = 3$
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner.
- At each time, we should have a partitioning of the appeared nodes into $k$ clusters.
- We are allowed to **merge clusters** but we cannot break them!
Online Clustering Problem

Online Clustering

- The set of points appear in an online manner
- At each time, we should have a partitioning of the appeared nodes into \( k \) clusters.
- We are allowed to **merge clusters** but we cannot break them!
In the next class, we study a clustering algorithm which uses a bidding algorithm as a black box!