COMP 7720 - Online Algorithms
Online Clustering & List Update

Shahin Kamali

Lecture 4 - Sep. 18, 2018
University of Manitoba
Review & Plan
Today’s objectives

- Online Clustering problem:
  - How to reduce an online problem to another
- List Update problem
Online betting: a Review

- 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 $u$.
  - The cost of the algorithm is $d_0 + d_1 + \ldots + d_k$
  - We have $d_0 < d_1 < \ldots < d_{k-1} < u \leq d_k$.

Magical betting formula

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

- When bids are 1, 2, $\ldots$, $2^i$, the competitive ratio is 4, and it is the best that a deterministic algorithm can do.
- If we select $X$ randomly from $U[0, 1)$ and use bids $e^X, e^{X+1}, \ldots, e^{X+k}$, the competitive ratio becomes $e \approx 2.71$, and it is the best a randomized algorithm can do.
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
- The objective is to achieve a clustering with minimum diameter.
- The problem is NP-hard (what does it mean?)
- Assume $k = 3$
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 divide one.
- Assume $k=3$
Greedy Algorithm: a First Approach

- Place the first $k$ points in $k$ different clusters.
- Greedily place any incoming point into the closest cluster!
- Is this algorithm competitive?
  - Assume the first $k$ points are at distance at most 1 from each-other and the next point is at distance $d$ of closest point, where $d$ is arbitrary large!.
  - Competitive ratio becomes at least $d$.
- A competitive online algorithm needs a mechanism to merge clusters!

![Diagram showing points and distances](image-url)
Clustering Algorithm: a Better Approach

- The algorithms uses a sequence $d_0, d_1, \ldots$ each associated with a phase.
- Each cluster is recognized by a center.
- At phase $i$, the distance between any pair of centers is more than $d_i$. 
Assume we are at phase $i$ and a new point $p$ arrives:

- If distance of $p$ to any center $c_i$ is at most $d_i$, add $P$ to the cluster of $c_i$.
- Else if there are fewer than $k$ clusters, create a new one for $P$.
- Otherwise, start phase $i + 1$
Starting a New Phase

- Create a temporary \((k + 1)\)th cluster with point \(P\).
- Process centers one by one: when processing center \(c_i\), merge its cluster with any cluster whose center is within distance \(d_{i+1}\) from \(c_i\).
  - If no merger occurred, go to the next phase.
Analysis

- Assume we are at phase $i + 1$
- At the beginning of phase $i + 1$, there has been $k + 1$ ‘centers’ (including temporary one)
  - Their pairwise distance is at least $d_i$.
- So, the cost of $\text{OPT}$ is at least $d_i$ (why?)
  - There are $k + 1$ centers of pairwise distance at least $d_i$; two of them have to be in the same cluster in any solution; such cluster will have diameter at least $d_i$. 
The radius of a cluster: max. distance of any point to the center
- Diameter is at most twice the radius.

When we merge other clusters to cluster $C$ at the beginning of the phase, the maximum radius is increased by at most $d_{i+1}$.
- The diameter is increased by at most $2d_{i+1}$.
- The diameter of any cluster at phase $i + 1$ is at most $2d_0 + 2d_1 + \ldots + 2d_{i+1}$. 
At any phase $i + 1$, the cost of $\text{OPT}$ is at least $d_i$ and the cost of the algorithm is at most $2(d_0 + d_1 + \ldots + d_{i+1})$.

The competitive ratio is at most $\frac{2(d_0 + d_1 + \ldots + d_{i+1})}{d_i}$

How to set $d_1, \ldots, d_{i+1}$ so that the above ratio is minimized?

- This is online bidding problem!

If we use doubling we get a competitive ratio of at most 8.

Randomized algorithm gives a competitive ratio of at most $2e \approx 5.4$. 
Concluding Remarks

- Any $c$-competitive algorithm for online bidding can be used to solve the online clustering algorithm.
  - The competitive ratio of such algorithm would be at most $2c$.
  - We can get competitive ratios of at most 8 and $2e$ respectively with doubling and randomized algorithm.

- Recall that the offline problem is NP-hard!
  - Without knowing the offline solution, we achieve online algorithms which guarantee they are no more than 8 (or $2e$) times worst that the optimal offline algorithm.
There are many variants of the clustering problem:

- Minimize the sum of diameters instead of maximum diameter.
- Minimize the number of clusters assuming the diameter cannot be more than a given value $D$.
- Consider a graph instead of plane!
  - Graph partitioning!

Potential topic for project: If you like geometry, consider variants settings for clustering (e.g., different objectives and different dimensions), specially under new models such as advice.
List Update Problem
List Accessing Problem

- The input is a set of requests to items in a list.
- The cost of accessing an item in index $i$ is $i$.

```
< d b b d c a c >
cost: 4+2+2+4+3+1+3 = 19
```

```
    a  b  c  d  e
```

- 4+2+2+4+3+1+3 = 19
Introduction to List Update

- An instance of self-adjusting data structures.
- The structure adjusts itself based on the input queries.
- List update was formulated in 1984 by Sleator and Tarjan
  - This result of Sleator and Tarjan made online algorithms popular in the following two decades
  - There are applications in data-compression!
Self-Adjusting Lists

- Update a list of length $k$ to adjust it to the patterns in the input sequence of length $n$ ($n \gg k$).
  - Free exchanges: Move a requested item closer to the front without any cost.
  - Paid exchanges: Swap positions of two consecutive items with a cost 1.

$$<d\ b\ b\ d\ c\ a\ c>$$

Cost: 4
List Update Problem

- In the offline version of the problem, you have access to the whole set at the beginning.
  - The problem is NP-hard.
- In the online setting, the requests appear in an online, sequential manner.
  - An online algorithm should reorder the list without looking at the future requests.
Move-To-Front (MTF)

- After each access, move the requested item to the front.
- It only uses free exchanges.

\[
\begin{align*}
\langle d \ b \ b \ d \ c \ a \ c \rangle \\
\text{cost: } & 4+3+1+2+4+4+2
\end{align*}
\]
After an access to $x$, move $x$ to the front of the first item $y$ which has been requested at most once since the last access to $x$.

Do nothing if such an item $y$ does not exist.

$< \text{d b b d c a c} >$

Cost: $4 + 2 + 2 + 4 + 4 + 3 + 4$
Optimal Static Algorithm

- Look at the sequence of requests, sort items by the frequency of their accesses.
  - The most accessed item will be at the beginning of the list.
- The cost of the algorithm would be at most $nk/2$. 
Consider a \textit{cruel} sequence in which the adversary always asks for the last item in the list!

What will be the cost of the algorithm?
- It will be $nk$.

What is the cost of $\text{OPT}$?
- We know the optimal static algorithm has a cost of $nk/2$.
- So the cost of $\text{OPT}$ is no more than $nk/2$.

The competitive ratio of any online list update algorithm is at least $\frac{nk}{nk/2} = 2$. 
Analysis of MTF

- In the next class, we learn that the competitive ratio of MTF is 2, i.e., it is the optimal deterministic algorithm for list update!
- We also learn about randomized list update algorithms.