Graded Homework 2, exercise 2

Marco Milanta

December 13, 2021

For this exercise I exchanged ideas with Jan Tempus.

Minimizing Cost (25 points)

Given a stream of n elements $x_1, \cdots, x_n \in \{1, \cdots, n\}$ all distinct (i.e., the input is a permutation). We have to choose x_i, x_j with $j \geq i$ on arrival, meaning that when an element arrives, we must immediately decide if we include it in one of the two elements. The cost function is given as $\frac{1}{x_j - x_i + 1}$. We would like to minimize this cost. (Note that any sensible strategy will have $\frac{1}{x_j - x_i + 1} < 1$ as we can choose $i = j = n$.)

1. Find a deterministic algorithm that achieves competitive ratio at most $\sqrt{n}$.
2. Show that any randomized algorithm has competitive ratio at least $\Omega(\sqrt{n})$

Solutions

1. We propose the following algorithm. Note that we index arrays starting from 1.

Legality of the algorithm. Before analyzing the competitive ratio of this algorithm, I would like to notice that it has the correct structure. Notice these two facts:

- In the $k-$th iteration of the loop, we only look at $X[k]$. This means that our algorithm doesn't cheat by looking into the future. Finally, we look at the last element after the loop.
- x_i and x_j are assigned only once, always x_i before x_j, and they are assigned to be $X[k]$ at the $k-$th iteration. Or, at latest, after the for loop. All of this is guaranteed by the counter `ChosenElements`. This means that we are respecting the constraint "when an element arrives, we must immediately decide if we include it in one of the two elements".
- The algorithm iterates over the input elements, therefore is polynomial in n.

Competitiveness of the algorithm. Now we want to analyze the performance. To do so, let $\hat{i}, \hat{j}, \hat{i} \leq \hat{j}$ be an optimal solution that minimize $\frac{1}{x_{\hat{j}} - x_{\hat{i}} + 1}$. We call OPT the `cost` of the optimal algorithm. Notice that the optimal cost cannot go lower $\frac{1}{n}$, this is achieved when we manage to get $x_i = 1, x_j = n$.

Theorem 1 (Algorithm 1 is $\sqrt{n}$-competitive)**.** *The algorithm 1 is $\sqrt{n}$-competitive against the optimal solution. Or, formally, let $i, j,$ `cost` be the output of the algorithm 1. Let OPT be the maximal profit, then, for any input*

$$\texttt{profit} \leq \sqrt{n} OPT.$$

Proof. We divide the proof in two parts depending on the behavior of the algorithm.

Algorithm 1: Online algorithm

Input: $X \in$ permutations of $\{1, \ldots, n\}$

Output: $i, j, \frac{1}{x_j - x_i + 1}$

1 `Seen` $\leftarrow [0, \ldots, 0] \in \mathbb{N}^n$
2 `Missing` $\leftarrow [1, \ldots, n] \in \mathbb{N}^n$
3 `ChosenElements` $\leftarrow 0$
4 $x_i \leftarrow x_j \leftarrow 0$
5 **for** $k \in 1, \ldots, n-1$ **do**
6   **if** `ChosenElements` $= 0$ **then**
7   `Missing`$[X[k]] \leftarrow 0$
8   `optFutu` $\leftarrow$ max `Missing`
9   **if** (`optFutu` $- X[k] + 1) \geq \sqrt{n}$ **then**
10             $x_i \leftarrow X[k]$
11             $i \leftarrow k$
12   `ChosenElements` $\leftarrow 1$
13   **else if** `ChosenElements` $= 1$ **then**
14   **if** $X[k] =$ max `Missing` **then**
15             $x_j \leftarrow X[k]$
16             $j \leftarrow k$
17   `ChosenElements` $\leftarrow 2$
18 **if** `ChosenElements` $= 0$ **then**
19     $x_i \leftarrow x_j \leftarrow X[n]$
20     $i \leftarrow j \leftarrow n$
21 return $i, j, \frac{1}{x_j - x_i + 1}$

Case 1: (there is a $k \in 1, \ldots, n-1$ for which we enter the **if** at line 9) Let for now k be the iteration in which we enter the **if** at line 9. One can notice that we will end up with a final cost of $\frac{1}{\texttt{optFutu} - X[k] + 1}$. This is because we chose $x_i = X[k]$, and then we chose x_j to be the maximal value among the remaining ones. We can safely find such value since we are keeping track of the missing ones (`optFutu`). Once max `Missing` will arrive, we pick it for x_j. Notice that we cannot miss it since we are only looking at permutations as inputs. Hence, our algorithm has a cost of

$$\texttt{cost} \frac{1}{x_j - x_i + 1} = \frac{1}{\texttt{optFutu} - X[k] + 1}.$$

Using this, we get that:

$$\texttt{cost} = \frac{1}{\texttt{optFutu} - X[k] + 1} \overset{(i)}{\leq} \frac{1}{\sqrt{n}} = \sqrt{n}\frac{1}{n} \overset{(ii)}{\leq} \sqrt{n}OPT.$$

Where (i) follows from the fact that we enter the **if** at line 9, and therefore $\texttt{optFutu} - X[k] + 1 \geq \sqrt{n}$. (ii) follows from the fact that minimum possible cost is $\frac{1}{n}$. In this scenario we have shown that the algorithm is $\sqrt{n}$ competitive.

Case 2: (we never enter the **if** at line 9) In this scenario, our algorithm picks $x_i = x_j = n$, and our `costs` will be 1. We now want to show that $OPT \geq \frac{1}{\sqrt{n}}$.

To do so, we continue by contradiction. Assume that the optimal solution takes $\hat{i} \leq \hat{j}$ such that $OPT = \frac{1}{x_{\hat{j}} - x_{\hat{i}} + 1} < \frac{1}{\sqrt{n}}$. This yields that

$$x_{\hat{j}} - x_{\hat{i}} + 1 > \sqrt{n}.$$

Now we look at the behavior of our algorithm in this setting. Let's observe the moment when it gets to step $k = \hat{i}$. In this step $\texttt{optFutu} = \max \texttt{Missing} \geq x_{\hat{j}}$ (this is true since $\hat{j} > \hat{i}$, it's not equal, otherwise OPT cost would be 1). This yields that $\texttt{optFutu} - X[\hat{i}] + 1 \geq x_{\hat{j}} - x_{\hat{j}} + 1 > \sqrt{n}$, which itself it yields that we enter the **if** at line 9. This is however a contradiction.

From this we have that $OPT \geq \frac{1}{\sqrt{n}}$, and we get:

$$\texttt{cost} = 1 = \sqrt{n}\frac{1}{\sqrt{n}} \leq \sqrt{n} OPT.$$

Conclusion: We have shown that independently of which part of the algorithm triggers, we can guarantee that it outputs a $\sqrt{n}$-competitive solution □

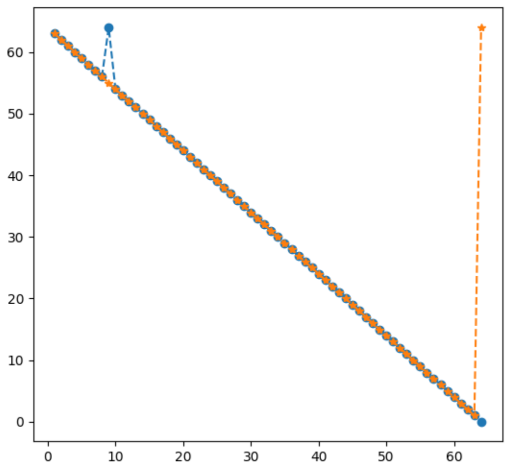
2. Let $OPT(x)$ be the optimal $\texttt{cost}$ achievable given $x \in \mathcal{X}$, where $\mathcal{X}$ is the space of permutation of $\{1, \dots, n\}$. Let's define a distribution over inputs $\mathcal{P}$ indexed by a discrete density $p : \mathcal{X} \to [0, 1]$:

$$p(x) = \begin{cases} \frac{1}{2} & x = \bar{x}_1 \\ \frac{1}{2} & x = \bar{x}_2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Where $\bar{x}_1, \bar{x}_2$ are just aliases for

$$\begin{aligned} \bar{x}_1 &= \left\{n-1, n-2, \dots, 1+n-\sqrt{n}, n, n-\sqrt{n}, n-\sqrt{n}-1, \dots, 1\right\} \\ \bar{x}_2 &= \{n-1, n-2, \dots, 1, n\}. \end{aligned}$$

Notice that we are assuming that $\sqrt{n}$ is an integer (we don't generalize here, but we are sure that the idea generalizes as well). To visualize them better, look at figure below.



Where $\bar{x}_1$ is represented with blue dots and $\bar{x}_2$ with orange stars. In this plot $n = 64$.

Now we state a key lemma that shows the importance of the chosen $\mathcal{P}$.

Lemma 1. *[Best distribution] Given $x \sim \mathcal{P}$, any possible deterministic algorithm $\mathcal{A}$ has an expected competitive ratio of at least $\frac{1}{2}(\sqrt{n}+1)$. Formally we have that,*

$$\mathbb{E}_{x\sim\mathcal{P}}\left[\frac{\mathcal{A}(x)}{OPT(x)}\right] \geq \frac{1}{2}\left(\sqrt{n}+1\right) \qquad \textit{for any } \mathcal{A} \textit{ deterministic algorithm}$$

Proof. We start by defining two deterministic algorithms: $\mathcal{A}_1$ and $\mathcal{A}_2$.

- $\mathcal{A}_1$ waits all the way to $1+n-\sqrt{n}$, then buys $x_i = 1+n-\sqrt{n}$. $\mathcal{A}_1$ then waits for n to appear and then picks it (note that it will always appear).
- $\mathcal{A}_2$ commits and wait all the way to the second to last element, and then picks it. $\mathcal{A}_2$ then picks the last element if n is the only missing element, and it buys the second to last again if the only missing element is 1.

$\mathcal{A}_1$ and $\mathcal{A}_2$ competitive ratio: looking at the behavior of the algorithms, and at how $\bar{x}_1$ and $\bar{x}_2$ are defined, we can compute the cost of the optimal algorithm, of $\mathcal{A}_1$ and of $\mathcal{A}_2$ on the inputs $\bar{x}_1$ and $\bar{x}_2$:

$$\begin{aligned}
OPT(\bar{x}_1) &= \frac{1}{n-1-n-\sqrt{n}+1} = \frac{1}{\sqrt{n}} \\
OPT(\bar{x}_2) &= \frac{1}{n-1+1} = \frac{1}{n} \\
\mathcal{A}_1(\bar{x}_1) &= \frac{1}{n-1-n-\sqrt{n}+1} = \frac{1}{\sqrt{n}} \\
\mathcal{A}_1(\bar{x}_2) &= \frac{1}{n-1-n-\sqrt{n}+1} = \frac{1}{\sqrt{n}} \\
\mathcal{A}_2(\bar{x}_1) &= \frac{1}{n-1-(n-1)+1} = 1 \\
\mathcal{A}_2(\bar{x}_2) &= \frac{1}{n-1+1} = \frac{1}{n}.
\end{aligned}$$

Now we can compute:

$$\begin{aligned}
\mathbb{E}_{x\sim\mathcal{P}}\left[\frac{\mathcal{A}_1(x)}{OPT(x)}\right] &= p\left(\bar{x}_1\right)\frac{\mathcal{A}_1(\bar{x}_1)}{OPT(\bar{x}_1)} + p\left(\bar{x}_2\right)\frac{\mathcal{A}_1(\bar{x}_2)}{OPT(\bar{x}_2)} \\
&= \frac{1}{2}\left(\frac{\frac{1}{\sqrt{n}}}{\frac{1}{\sqrt{n}}} + \frac{\frac{1}{\sqrt{n}}}{\frac{1}{n}}\right) = \frac{1}{2}\left(1+\sqrt{n}\right)
\end{aligned}$$

And we also do it for $\mathcal{A}_2$:

$$\begin{aligned}
\mathbb{E}_{x\sim\mathcal{P}}\left[\frac{\mathcal{A}_2(x)}{OPT(x)}\right] &= p\left(\bar{x}_1\right)\frac{\mathcal{A}_2(\bar{x}_1)}{OPT(\bar{x}_1)} + p\left(\bar{x}_2\right)\frac{\mathcal{A}_2(\bar{x}_2)}{OPT(\bar{x}_2)} \\
&= \frac{1}{2}\left(\frac{1}{\frac{1}{\sqrt{n}}} + \frac{\frac{1}{n}}{\frac{1}{n}}\right) = \frac{1}{2}\left(\sqrt{n}+1\right)
\end{aligned}$$

Any other algorithm: Now we want to show that the result doesn't only hold fore $\mathcal{A}_1$ and $\mathcal{A}_2$, but also for any other deterministic algorithm. For this we group all the possible algorithms in 2 groups:

- **Group 1**: algorithms that pick as first element an element before $1+n-\sqrt{n}$ both in $\bar{x}_1$ and $\bar{x}_2$
- **Group 2**: Every algorithm which is not in **Group 1**, and it's not $\mathcal{A}_1$ or $\mathcal{A}_2$.

First we show that any algorithm in **Group 1** costs more than $\mathcal{A}_1$ independently of the input $\bar{x}_1$ and $\bar{x}_2$. Let $\mathcal{C}$ be an algorithm in **Group 1**: it picks the first element c_1 before $1+n-\sqrt{n}$, and then it picks c_2. This implies

$$\mathcal{C}(\bar{x}_1) = \mathcal{C}(\bar{x}_2) = \frac{1}{c_2-c_1+1} \overset{(i)}{<} \frac{1}{c_2-(1+n-\sqrt{n})+1} \overset{(ii)}{\leq} \frac{1}{n-1-n+\sqrt{n}+1} = \frac{1}{\sqrt{n}}$$

Where the inequality (i) holds since we c_1 is before $1+n-\sqrt{n}$, and therefore $c_1 > 1+n-\sqrt{n}$. (ii) holds since $c_2 \leq n$. Notice that both in $\bar{x}_1$ and $\bar{x}_2$ the cost of $\mathcal{C}$ is more than $\frac{1}{\sqrt{n}} = \mathcal{A}_1(\bar{x}_1) = \mathcal{A}_1(\bar{x}_2)$. Therefore:

$$\mathbb{E}_{x\sim\mathcal{P}}\left[\frac{\mathcal{C}(x)}{OPT(x)}\right] \geq \mathbb{E}_{x\sim\mathcal{P}}\left[\frac{\mathcal{A}_1(x)}{OPT(x)}\right] \geq \frac{1}{2}(\sqrt{n}+1).$$

This concludes the proof for **Group 1**.

We now notice that all the algorithms in **Group 2** buy the first element after $1+n-\sqrt{n}$. This is not trivial since in **Group 1** there are the algorithms that pick the first element before $1+n-\sqrt{n}$ both in $\bar{x}_1$ and $\bar{x}_2$. What about the algorithms that depending on the input chose to pick either before or after $1+n-\sqrt{n}$? Such algorithm cannot exist among deterministic ones, this is because the first elements before $1+n-\sqrt{n}$ are exactly the same in $\bar{x}_1$ and $\bar{x}_2$, therefore, a deterministic algorithm that picks an element before $1+n-\sqrt{n}$ will do it in both inputs, and vice versa, if it doesn't pick it before $1+n-\sqrt{n}$ it will not do it in both inputs.

Now, once we observed this, it is easy to notice that all algorithms in group to get a cost of at least 1 on the input $\bar{x}_1$, this is because they "missed" the spike at $1+n-\sqrt{n}$.

Now let $\mathcal{B}$ be our algorithm in **Group 2**. Let c_1 be the element $\mathcal{B}$ picks first on the input $\bar{x}_2$, and c_2 the other. $c_1 > 1$, otherwise the algorithm is $\mathcal{A}_2$, but **Group 2** doesn't contain $\mathcal{A}_2$. And, as before, $c_2 \leq n$. This yields that the cost in input $\bar{x}_2$ is

$$\mathcal{B}(\bar{x}_2) = \frac{1}{c_2-c_1+1} \geq \frac{1}{n-c_1+1} > \frac{1}{n-1+1} = \frac{1}{n}.$$

Furthermore, from the observation before, we have $\mathcal{B}(\bar{x}_1) \geq 1$. Finally, we can observe that $\mathcal{B}$ does strictly worse than $\mathcal{A}_2$ both in $\bar{x}_1$ and $\bar{x}_2$. Therefore

$$\mathbb{E}_{x\sim\mathcal{P}}\left[\frac{\mathcal{B}(x)}{OPT(x)}\right] \geq \mathbb{E}_{x\sim\mathcal{P}}\left[\frac{\mathcal{A}_2(x)}{OPT(x)}\right] \geq \frac{1}{2}(\sqrt{n}+1).$$

Finally, this concludes the proof also for **Group 2**. It's easy to see that any algorithm is either $\mathcal{A}_1, \mathcal{A}_2$, inside **Group 1** or inside **Group 2**, hence we have concluded the proof. □

Now, the result simply follows from Yao's principle,

$$\max_{x\in\mathcal{X}} \frac{\mathbb{E}_{a\sim\mathcal{A}}[a(x)]}{OPT(x)} \overset{(i)}{=} \max_{x\in\mathcal{X}} \mathbb{E}_{a\sim\mathcal{A}}\left[\frac{a(x)}{OPT(x)}\right] \overset{(ii)}{\geq} \min_{\mathcal{A}\in\text{det. algos}} \mathbb{E}_{x\sim\mathcal{P}}\left[\frac{\mathcal{A}(x)}{OPT(x)}\right] \overset{(iii)}{\geq} \frac{1}{2}(\sqrt{n}+1) \in \Omega(\sqrt{n})$$

Where (i) follows from linearity of expectation, (ii) is a direct application of Yao's principle using $c(a,x) = \frac{a(x)}{OPT(x)}$. (iii) follows from Lemma 1. Being the first term of the last equation the competitive ratio of an arbitrary randomized algorithm $\mathcal{A}$, we have showed that it is in $\Omega\left(\sqrt{n}\right)$, and therefore, we have concluded the proof.