Release: September 1, 2020: 12:00 Deadline: September 2, 2020: 12:00

Time: 24 hours. You may refer to all resources available to you, but discussion is strictly prohibited. Submission format: PDF is preferred. Photographs and scans are also acceptable, but it's your responsibility to ensure that they are clear and readable.

1. Here is a simplified version of the problem which food delivery companies like Swiggy face. Let us call it the k-Swiggy problem. Like in the k-server problem, k delivery persons occupy points in a metric space. Each request consists of a (restaurant,customer) pair. To serve the request, Swiggy sends one delivery person to the restaurant, who picks up the order and delivers it to the customer. Let us assume that Swiggy gets the next request only after it has served the current request, so all the k persons are available to serve any request.

   Mathematically, each request i is a pair of points (s_i, t_i) from the metric space. Such a request is served by moving a server to s_i and then moving the same server from s_i to t_i. The cost incurred is, naturally, the total distance travelled by the k servers (inclusive of the distances from s_i to t_i). Let α_k and β_k denote the deterministic competitive ratios of the k-server and the k-Swiggy problem respectively. (Recall that $k \leq \alpha_k \leq 2k-1$.)

   (a) **[3 points]** Prove that $\alpha_k \leq \beta_k$. (That is, show how you can use any online k-Swiggy algorithm for the k-server problem.)

   (b) **[7 points]** Prove that $\beta_k \leq \alpha_k + 2$. (That is, show how you can use any online k-server algorithm to get a k-Swiggy algorithm while adding at most 2 to the competitive ratio.)

2. Recall the problem of pre-emptive online matching in the edge-arrival setting. Edges arrive one after the other. We can accept an edge only as soon as it comes, and we can discard it in future. We must always maintain a matching M in the graph. To prove competitiveness, we showed that we can maintain values x_v for each vertex v that constitute a feasible dual solution, such that for every new edge arrival, the change in the algorithm's matching's weight is at least the competitive ratio times the change in dual.

   (a) **[7 points]** We call an instance *b-special* if the ratio of the weights of any two edges is an integer power of b, for some parameter $b > 2$. Consider the following algorithm, which we will call algorithm $\mathcal{A}$. Start with $M = \emptyset$. For every new edge e with weight w, if w is greater than the weight of every conflicting edge in M, then remove those conflicting edges from M and add e to M. Else, discard e. Explain how you will maintain the dual variables to prove that on b-special instances, this algorithm has competitive ratio at least

   $$\frac{b-2}{2 \cdot (b-1)}.$$

   (b) Let us now analyze a randomized algorithm for pre-emptive online matching. For a fixed $b > 2$, define $f(w, r)$ to be the largest number less than or equal to w that is of the form b^{z+r} for some integer z (equivalently, $\log_b f(w, r) = \lfloor \log_b w - r \rfloor + r$). The algorithm $\mathcal{A}'$ is as follows. Choose r uniformly at random from $[0, 1]$. For every edge of weight w, reassign its weight to be $f(w, r)$ and pass it to algorithm $\mathcal{A}$. In the end, simply return the output of $\mathcal{A}$. Observe that the instance received by $\mathcal{A}$ is b-special, so the guarantee from the previous question applies.

      i. **[6 points]** Prove that

      $$\mathbb{E}_{r \sim U[0,1]}[f(w, r)] = w \cdot \frac{b-1}{b \ln b},$$

      where $U[0, 1]$ denotes the uniform distribution on $[0, 1]$. (Hint: How is $\log_b f(w, r)$ distributed?)

      ii. **[7 points]** Now prove that algorithm $\mathcal{A}'$ has competitive ratio at least

      $$\frac{b-2}{2 \cdot (b-1)} \times \frac{b-1}{b \ln b} = \frac{b-2}{2b \ln b}.$$

You may assume that all the earlier claims are true, even if you couldn't prove them. (It is easy to see that the maximum value of the competitive ratio is attained when b satisfies $2 \ln b = b - 2$, and the maximum value is ≈ 0.1867, better than the deterministic competitive ratio.)

3. The goal of this question is to analyze another algorithm for the prophet problem. Assume that none of the independent non-negative random variables $X_1, \ldots, X_n$ have point masses, that is, their CDFs are all continuous. Their realizations are seen in the order $X_1, \ldots, X_n$. Let the threshold T be such that $\sum_{i=1}^{n} \Pr[X_i \geq T] = 1$. (Such a T exists because the CDFs are continuous.) Define the random variables $\overline{X}_1, \ldots, \overline{X}_n$ as $\overline{X}_i = X_i \cdot \mathbb{I}[X_i \geq T]$, (that is, $\overline{X}_i = X_i$ if $X_i \geq T$ and $\overline{X}_i = 0$ otherwise).

   (a) **[4 points]** Prove that $\mathbb{E}[\max_i X_i] \leq \sum_{i=1}^{n} \mathbb{E}\left[\overline{X}_i\right]$.

   (b) **[6 points]** Define $p_i = \Pr[X_i \geq T]$. Consider the following algorithm. For $i = 1$ to n, on receiving X_i:

   - If $X_i \geq T$, then with probability $1/(2 - \sum_{j<i} p_j)$ accept X_i and stop, and with probability $1 - 1/(2 - \sum_{j<i} p_j)$ reject X_i and continue.
   - Else (i.e. if $X_i < T$), reject X_i and continue.

   (Note that $1/(2 - \sum_{j<i} p_j)$ is a valid probability because $\sum_{j<i} p_j \leq \sum_{j=1}^{n} p_j = 1$.) Determine the probability that the algorithm doesn't stop before it sees X_i. Hence, prove that the expected reward of this algorithm is $\left(\sum_{i=1}^{n} \mathbb{E}\left[\overline{X}_i\right]\right)/2$. (Therefore, this algorithm too is (1/2)-competitive.)