TDA251
Home Exam

Wang QuFei
900212-6952
qufei@student.chalmers.se

January 15, 2020

1 Algorithm Design

1.1 A Randomized Algorithm

A randomized algorithm can be described as follows. Before going into the details of the algorithm, we first claim that any instance of the problem can be transformed into a special case where two dots at position 0 and 1 can be found. In fact, for any instance we can probe first at position 0 with result l_1, r_1, then probe at position 1 second with result l_2, r_2. Denoted l and r as the coordinates of the dots on the edge of the instance. If $l \neq 0$ or $r \neq 1$, by resizing the interval from $[0, 1]$ to $[l, r]$, and calculating the corresponding minimum distance d^* by $d^* = d(r - l)$, we get an instance with the special form we claimed, with the number of dots unchanged.

Suppose the number of probes available is m, $m \ll n$, then the algorithm proceeds as:

1. mark the interval $[0, 1]$ as 'unprobed'
2. maintain a sorted list to remember the segments discovered so far. The segments are sorted by their length on ascending order
3. for $i = 1 \dots m$, probe uniformly at random on all 'unprobed' parts on the interval with result l, r
   (a) mark segment $[l, r]$ as 'probed'
   (b) insert (l, r) into the sorted list
4. for all the parts of the interval [0 ,1] that have not been probed, each denoted as u_j
   (a) find all the discovered segments from the sorted list with length smaller or equal to $length(u_j)$
   (b) if the number of such segmetns is not zero, calculate the average distance of these discovered segments, denoted as d_{u_j}, estimate the number of segments u_j contains by $n_{u_j} = \lfloor length(u_j)/d_{u_j} \rfloor$
   (c) Otherwise if no segment discovered has length smaller or equal to $length(u_j)$, let $n_{u_j} = 1$
5. return $m + \sum_j n_{u_j} + 1$ as the estimated result

The idea behind this algorithm is that, if we classify the $n-1$ segments into different types i, where the segment s in type i satisfies $length(s) \in [id, (i+1)d)$. Then the type of segments with larger total length have higher probability to be choosen during the process of the algorithm. When we estimate some part of the interval [0, 1] left undiscovered, it is reasonable to infer its composition, namely the number of different kinds of segments it may contain, by the information of the segments with equal or smaller length we've already got, which is expressed by the average of their distance. Notice that for a given instance of this problem with certian segments composition, the probability of each type of segments being choosen is independent of their distribution.

2 Algorithm Analysis

2.1 Approximation Ratio and Probability

Let n^* be the estimated result. The bound on the approximation ration of $n^* : n$ can be analyzed as follows:

1. to find a upper bound on the approximation ratio, imagine there are m segments in [0, 1] with length d and they are choosen out by the m probes. In this case, $n^* : n = \frac{m+(1-md)/d+1}{n}$
2. to find a lower bound on the approximation ratio, imagine the $(n-1-m)$ segments take a total length $(n-1-m)d$, the remaining $1-(n-1-m)$ part of the interval are occupied by m segments, each with a length larger than $(n-1-m)d$, in this case, $n^* : n = \frac{m+2}{n}$

To consider the probability of large deviation of our estimation result from n, we should first be aware of how large a deviation can actually be. Notice that for an instance of the problem where all dots are evenly distributed, the result of our algorithm is exactly n. In this case, the probability of any deviation from n is zero. Therefore, instead of seeking for an answer with regard to the following general formula,

$$H(m) = Pr(|m + \sum_j n_{u_j} + 1 - n| \geq \Delta \cdot n)$$

we define the 'large deviation' as the worst case scenario for a given instance, where the algorithm selects the m smallest segments or m largest segment so as to produce a deviation as far as it can be.

The follwing analysis is similar with the approximation ratio we did above. For a given instance of this problem, let $seg_1, \ldots, seg_m$ be the m smallest segments with ascending order of length. Define F_i as the event one of these segments is selected in the ith probe. Our goal is to find $Pr(F_1 \cap F_2 \cdots \cap F_m)$. Let α_i be the length of seg_i, for the m segments we consider, there are $m!$ cases where they are choosen as the m probes we describe. Among these $m!$ events, the one with the largest probability would be to choose from seg_m down to seg_1. This is intuitively reasonable. Thus we have

$$Pr(F_1 \cap F_2 \cdots \cap F_m) \leq m! \cdot \alpha_m \cdot \frac{\alpha_{m-1}}{1-\alpha_m} \cdot \cdots \cdot \frac{\alpha_1}{1-\sum_{i=2}^{m} \alpha_i}$$

Similiarly, for the $seg'_1, \ldots, seg'_m$ be the m largest segments in descending order, we can bind the probability on the event 'the largest m segments are selected in the m probes' as

$$Pr(F'_1 \cap F'_2 \cdots \cap F'_m) \leq m! \cdot \alpha'_1 \cdot \frac{\alpha'_2}{1-\alpha'_1} \cdot \cdots \cdot \frac{\alpha'_m}{1-\sum_{i=1}^{m-1} \alpha'_i}$$

where F'_i denotes one of these segments is selected in the ith probe.

2.2 Any Deterministic Algorithm

We claim that no effcient deterministic algorithm exists for this problem.

Proof. For any arbitrarily bad(unevenly distributed) instance of this problem, the only way to approximate n using limited number of probes is to minimize the 'unprobed' space after the use of probe resources.The larger space the undiscovered segments left, the higher probability there exists for large estimation error. Since any 'deterministic algorithms can be defined in terms of a state machine' [1], with internal rules deciding how the inner states transition between each other . Suppose α is a deterministic algorithm for this problem, we could build an instance of the problem such that, under given input, for any transition rule $(s_i, (l_i, r_i)) \to s_{i+1}$ it defines, the next position it chooses to probe will always result in a segment with minimal length d. Since the problem itself does not provide any other way to get information about the segments, this algorithm will come to a halt after the exhaustion of all available resources, with only m segments of mininal size detected and large number of segments undiscovered.

2.3 Assumption of a Minimum Distance d

The assumption of a minimum distance d is crucial in that it is vital for the argument of the upper and lower bound on the estimation ratio described above. If we drop this assumption, although the randomized algorithm 1.1 can proceed as usual, we can no longer say anything about the bound on the estimation ratio, for any undiscovered segment may contain potentially unlimited number of dots.

2.4 A Randomized PTAS

A randomized PTAS approach can be achived by a slight modification of the randomized algorithm above. Given ϵ as the parameter ,

1. mark the interval $[0, 1]$ as 'unprobed'
2. maintain a sorted list to remember the segments discovered so far. The segments are sorted by their length on ascending order
3. probe uniformly at random on all 'unprobed' parts on the interval with result l, r
   (a) mark segment $[l, r]$ as 'probed'
   (b) insert (l, r) into the sorted list
   (c) denote $s_d = s_d + r - l$ as the sum of the distance measured
   (d) for all the parts in the interval [0 ,1] that have not been probed currently, denoted each as u_j

[1] `https://en.wikipedia.org/wiki/Deterministic_algorithm`

i. find all the discovered segments from the sorted list with length smaller or equal to $length(u_j)$

ii. if the number of such segmetns is not zero, calculate the average distance of these discovered segments, denoted as d_{u_j}, estimate the number of segments u_j contains by $n_{u_j} = \lfloor length(u_j)/d_{u_j} \rfloor$

iii. Otherwise if no segment discovered has length smaller or equal to $length(u_j)$, let $n_{u_j} = 1$

(e) denote $n' = m + \sum_j n_{u_j} + 1$ as the current estimation value

(f) let $n_{low} = m, n_{high} = m+(1-s_d)/d$, if $n_{low} \geq (1-\epsilon)n'$ and $(1+\epsilon)n' \leq n_{high}$, return n' as the estimated value. Otherwise, repeat (d).