On Density, Threshold and Emptiness Queries for Intervals in the Streaming Model

Arijit Bishnu[1], Amit Chakrabarti[2], Subhas C. Nandy[1], and Sandeep Sen[3]

1 **Indian Statistical Institute, Kolkata, India**
`{arijit,nandysc}@isical.ac.in`

2 **Department of Computer Science, Dartmouth College, Hanover, USA**
`ac@cs.dartmouth.edu`

3 **Department of Computer Science Engineering, Indian Institute of Technology – Delhi, New Delhi, India**
`ssen@cse.iitd.ernet.in`

Abstract

In this paper, we study the maximum density, threshold and emptiness queries for intervals in the streaming model. The input is a stream $\mathcal{S}$ of n points in the real line $\mathbb{R}$ and a floating closed interval W of width α. The specific problems we consider in this paper are as follows.

- Maximum density: find a placement of W in $\mathbb{R}$ containing the maximum number of points of $\mathcal{S}$.
- Threshold query: find a placement of W in $\mathbb{R}$, if it exists, that contains at least Δ elements of $\mathcal{S}$.
- Emptiness query: find, if possible, a placement of W within the extent of $\mathcal{S}$ so that the interior of W does not contain any element of $\mathcal{S}$.

The stream $\mathcal{S}$, being huge, does not fit into main memory and can be read sequentially at most a constant number of times, usually once. The problems studied here in the geometric setting have relations to frequency estimation and heavy hitter identification in a stream of data. We provide lower bounds and results on trade-off between extra space and quality of solution. We also discuss generalizations for the higher dimensional variants for a few cases.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Density, threshold, emptiness queries, interval queries, streaming model, heavy hitter, frequency estimation

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2015.336

1 Introduction

Motivated by problems related to chip density and thermal analysis in VLSI [4, 19], researchers in computational geometry have looked at problems involving windowing queries on a point set [5], such as maximum empty rectangle query [2] and maximum density query [23]. Windowing queries or their one-dimensional counterpart – interval queries – have motivations in geospatial and sensor network applications [16], where huge amounts of data are generated continuously in a stream, and communication is very expensive. Thus, it is preferable to communicate an appropriate summary of the data. Moreover, devices used for this purpose have limited memory. This calls for solving the problems on streaming data using limited amount of memory.

We consider density, threshold, and emptiness queries for a fixed-length interval among points in $\mathbb{R}$ in the streaming model [3, 22]. In the *pure* or *one-pass streaming model*, the data can be read only once and in the *multi-pass streaming model*, the data can be read more than once but always in the same order; in both cases the data is read-only. Apart from the number of passes, the other crucial issues in the streaming model are the amount of working memory used to process the input and the time taken, per item in the stream, to update this working memory. Ideally, both these quantities should preferably be significantly sub-linear in the size of the input.

1.1 The Computational Model and Problems Considered

Our input is a parameter $\alpha > 0$, representing the width of a *closed* interval W, plus a stream (i.e., sequence) $\mathcal{S} = \langle s_1, \ldots, s_n \rangle$ of n points, where each $s_i \in \mathbb{R}$. We consider certain interval queries that involve placing W suitably among the points in $\mathcal{S}$ so as to satisfy certain objectives. The combinatorial nature of these queries ensures that it suffices to consider only those placements where the left end-point of W coincides with a point in $\mathcal{S}$. Therefore, we define, for each $s \in \mathcal{S}$, the set $\mathcal{I}_s(\alpha) := \{x' \mid x' \in \mathcal{S} \text{ and } s \leqslant x' \leqslant s + \alpha\}$.

Our problems of interest are as follows.

- The *maximum density problem*, where the goal is to find $\max_{s \in \mathcal{S}} |\mathcal{I}_s(\alpha)|$ and a choice of s that achieves the maximum; the problem is denoted as MAX-DENSE.
- The *sorted-order* maximum density problem, denoted MAX-DENSE-SORTED, where the stream $\mathcal{S}$ arrives in a sorted fashion, i.e., $s_1 \leqslant s_2 \leqslant \cdots \leqslant s_n$ and the goal is the same as above.
- The *threshold query* problem, denoted THRESHOLD, where we must report whether there exists an s with $|\mathcal{I}_s(\alpha)| \geqslant \Delta$; the parameter Δ is known in advance.
- The *emptiness query* problem, denoted EMPTINESS, where we must report whether there exists an $s \in \mathcal{S} \setminus \{\max \mathcal{S}\}$ with $|\mathcal{I}_s(\alpha)| = 1$. This amounts to asking whether the *open* interval $\operatorname{int}(W)$ can be placed within $[\min \mathcal{S}, \max \mathcal{S}]$ so that it avoids all points in $\mathcal{S}$.

Computations take place in a word RAM with word size large enough to hold the parameters α and Δ, a single point s_i, and a $\lceil \log n \rceil$-bit counter. This essentially means that all "real numbers" appearing as inputs are in fact rationals with bounded bit precision. In our algorithms, the precision of intermediate computations will be within a constant factor of the word size. The stream length n may not be known *a priori*, but thanks to standard techniques we can often nevertheless design algorithms pretending that it is.

1.2 Our Results

We first observe that MAX-DENSE is hard: even a randomized algorithm that approximates $OPT := \max_s |\mathcal{I}_s(\alpha)|$ up to a large constant factor requires $\Omega(n)$ bits of space.[1] On the other hand, MAX-DENSE-SORTED admits a deterministic $(1-\epsilon)$-factor approximation[2] using $O(\epsilon^{-1} \log(\epsilon n))$ words of working space. Returning to MAX-DENSE, we show that given a Δ such that $OPT \geqslant \Delta$, we can compute a $(1-\epsilon)$-factor approximation with high probability using $O(\frac{n \log n}{\epsilon^3 \Delta})$ words. We also suggest a 3-factor approximation algorithm that runs in $O(n \log n)$ time. For the natural generalization of MAX-DENSE to $\mathbb{R}^2$, we show that a 6-approximation can be obtained in $O(n \log M)$ time using $O(M)$ words of space where M is the size of the maximum independent set of the copies of W positioned such that a specific (say

[1] As is common in streaming algorithms, lower bounds are expressed in bits and upper bounds in words.
[2] A γ-factor approximation algorithm means a placement of W at a point $s \in \mathcal{S}$, such that $OPT \geqslant |\mathcal{I}_s(\alpha)| \geqslant \gamma \cdot OPT$.

top-left) corner is at each point in $\mathcal{S}$. For THRESHOLD, we provide approximate deterministic as well as random sampling based algorithms, achieving space bounds of $O_\epsilon(n/\Delta)$[3]. We prove a randomized $\Omega(n/\Delta^2)$ lower bound, showing that this is nearly tight. For EMPTINESS, we prove a strong lower bound of $\Omega(n)$ in the randomized case and an especially strong lower bound of $\frac{1}{2}n - O(1)$ in the deterministic case.

1.3 Related Work

The need to process massive data has generated considerable interest in the data streaming model [15, 22]. The geometric problems we study here have close ties with frequency moment estimation [3] and heavy hitter identification [11]; see also [25] for a short summary. In the extensively-studied frequency moments problem, the stream $\mathcal{S}$ consists of n integers, each in $\mathcal{M} := \{1, 2, \ldots, m\}$ for some $m = n^{\Theta(1)}$. Let $f_i = |\{j \mid s_j = i\}|$ denote the frequency of i in $\mathcal{S}$, and define for each $k \geqslant 0$, $F_k := \sum_{i=1}^{m} f_i^k$. By convention, F_0 is the number of distinct elements and F_∞ is $\max_{i \in \mathcal{M}} f_i$. Up to logarithmic (in n and m) factors, the space complexity for approximating F_k is known to be $\Theta(1)$ for $0 \leqslant k \leqslant 2$ and $\Theta(n^{1-2/k})$ for $k > 2$ [3, 17, 7]. In particular, approximating F_∞ to a large constant factor, e.g., 100, requires $\Theta(m)$ space [26].

In frequency estimation, apart from the sequence $\mathcal{S}$, we have a threshold κ, $0 < \kappa < 1$ and we have to maintain an estimate $\bar{f}_i$, of f_i, such that $f_i - \kappa n \leqslant \bar{f}_i \leqslant f_i$. In heavy hitter identification, we have two parameters ξ and κ, $0 < \xi < \kappa < 1$, and we need to output all elements whose frequency is more than κn, and no element whose frequency is less than $(\kappa - \xi)n$ should be reported. Starting with the result of Misra and Gries [21], there have been a host of results [10, 12].

In the context of the present problem, Emek et al. [13] proposed a 2-factor approximation algorithm for computing the maximum independent set of intervals in streaming setup, that uses space linear in the size of the output. They also proposed a matching lower bound claiming that an approximation ratio of $2-\epsilon$ cannot be obtained by any randomized streaming algorithm with space significantly smaller than the size of the input (much larger than the output size). Recently, Cabello and Pérez-Lantero [6] showed that if the end-points of the intervals are in the set $\{1, 2, \ldots, n\}$, then an estimate $\hat{M}$ of the maximum independent set M can be obtained in space polynomial in ϵ^{-1} and $\log n$ which satisfies $\frac{1}{2}(1-\epsilon)M \leqslant \hat{M} \leqslant M$ with probability at least $\frac{2}{3}$. For equal length intervals, the estimate $\hat{M}$ of M satisfying $\frac{2}{3}(1-\epsilon)M \leqslant \hat{M} \leqslant M$ can be obtained using same amount of space satisfying the same probability bound.

In the multipass streaming model, there have been results related to approximate convex hull [16], approximate minimum enclosing ball [9]. Agarwal et al. [1] proposed a general technique for approximating various extent measures of a point set P in $\mathbb{R}^d$ in the streaming model. Chan [8] raised the issue of getting $O(1)$ space streaming algorithms for these problems. Apart from these, Har-Peled and Mazumdar [14] gave a $(1+\epsilon)$-approximation of the k-median and k-mean clustering of a stream of points in $\mathbb{R}^d$.

2 Interval Placement for Maximum Density

We start by observing that MAX-DENSE generalizes the problem of frequency moment estimation. Given a stream $\mathcal{S} = \langle s_1, \ldots, s_n \rangle$ of integers, we can consider its elements as points in $\mathbb{R}$. Take $\alpha = 0$. Then, for all $s \in \mathcal{S}$, the cardinality $|\mathcal{I}_s(\alpha)|$ is simply the frequency

[3] $O_\epsilon(f)$ denotes ϵ in the expression of f is treated as a constant.

of s. Therefore $\max_s |\mathcal{I}_s(\alpha)| = F_\infty(\mathcal{S})$. For the F_∞ problem, we recall an important result observed (though not formally written as a theorem) in Alon et al. [3]. Based on communication lower bounds for the multi-party SET-DISJOINTNESS problem, it follows that distinguishing $F_\infty(\mathcal{S}) = 1$ from $F_\infty(\mathcal{S}) \geqslant \Delta$ requires $\Omega(n/\Delta^2)$ space. This holds even for randomized and multipass streaming algorithms using $O(1)$ passes.

Based on the above, we obtain the following strong lower bound.

▶ **Theorem 2.1.** *For the* MAX-DENSE *problem, put* $OPT := \max_s |\mathcal{I}_s(\alpha)|$. *Any randomized constant-pass algorithm that distinguishes* $OPT \geqslant \Delta$ *from* $OPT = 1$ *with probability* $\geqslant 2/3$ *uses* $\Omega(n/\Delta^2)$ *bits of space. In particular, approximating* OPT *to any constant fraction requires* $\Omega(n)$ *space.*

If one wants to avoid degeneracy, a simple perturbation argument allows one to replace the hard instance for MAX-DENSE implicit above with one in which all the points in the stream are distinct and $\alpha > 0$.

2.1 Points in Sorted Order: The Problem max-dense-sorted

In light of Theorem 2.1, we consider the easier variant MAX-DENSE-SORTED, where the input stream satisfies $s_1 \leqslant s_2 \leqslant \cdots \leqslant s_n$. For this variant, we start by describing an output sensitive procedure, followed by a 2-approximation algorithm and finally, we give a very space-efficient deterministic algorithm achieving a $(1+\epsilon)$-approximation.

The optimum solution OPT can be computed using OPT counters using the following simple *output sensitive* procedure. We allocate counters to count points in $\mathcal{I}_{s_1}(\alpha), \mathcal{I}_{s_2}(\alpha), \ldots,$ until we get a point $p \in \mathcal{S}$ that lies outside $s_1 + \alpha$. At this point of time, the counter for $\mathcal{I}_{s_1}(\alpha)$ is closed, and a new counter for $\mathcal{I}_p(\alpha)$ is initiated. At any instant of time, the maximum of the contents of all closed counters is stored in OPT. At the end of the stream, all the active counters are closed, and we report the content of OPT. Thus the maximum number of counters active at any point of time is bounded by OPT. For n points uniformly distributed in the interval $[\ell, u]$, this algorithm will be very space efficient when $\frac{u-\ell}{\alpha} \geqslant n/polylog(n)$ – see Lemma 1.1 of the Appendix.

A simple 2-approximation algorithm is easy to obtain. (We would need this idea when we discuss THRESHOLD.) We initiate a counter for counting points in $\mathcal{I}_{s_1}(\alpha)$. The counting continues until we get a point $p \in \mathcal{S}$ that lies outside $[s_1, s_1 + \alpha]$. The same counter now starts counting for $\mathcal{I}_p(\alpha)$. Finally, report the interval with maximum number of points. The approximation ratio follows from the fact that the optimal α-interval spans on two consecutive α-intervals $\mathcal{I}_{s_i}(\alpha)$ and $\mathcal{I}_{s_{i+1}}(\alpha)$ for which we have computed the number of points.

Next, we discuss the $(1+\epsilon)$-approximation, the main result of this subsection. For our algorithm, we introduce a subroutine that we call *the* (k, B)*-process* (k and B are positive integers), defined as follows. We maintain up to B active counters for certain sets $\mathcal{I}_s(\alpha)$, plus a register $n_{\max}$ to record the maximum value ever seen in a counter. At every kth stream element s – i.e., for $s \in \langle s_1, s_{1+k}, s_{1+2k}, \ldots \rangle$ – first close all active counters for sets $\mathcal{I}_{s'}(\alpha)$ such that $s > s' + \alpha$, updating $n_{\max}$ as needed. Reclaim the space allocated to all closed counters. Then, if there is space, open a new counter to accurately count $\mathcal{I}_s(\alpha)$; if there is no room – i.e., we are about to open a $(B+1)$th counter – then abort the process instead. Increment all active counters. At the end of the stream, if we haven't aborted, output $n_{\max}$.

We make use of the simple observation that if the (k, B)-process aborts, then $OPT > kB$; otherwise, $OPT - k \leqslant n_{\max} \leqslant OPT$.[4]

[4] The initial idea of classifying OPT was conveyed to one of the authors by Sai Praneeth.

```
Algorithm 1: active-process $(k, B)$
  Input stream $s_1, s_2 \ldots$
  A set $\mathcal{C}$ of $B$ counters. Initialize the first counter for $\mathcal{I}_{s_1}(\alpha)$.   $n_{\max} \leftarrow 0$ ;
1 for $s \in \{s_{1+k}, s_{1+2k}, s_{1+ik} \ldots\}$ do
2     Close all active counters for $\mathcal{I}_{s'}(\alpha)$ for $s > s' + \alpha$ and update $n_{\max}$ ;
3     Increment all active counters (for $\mathcal{I}_{s'}(\alpha)$ for $s \leqslant s' + \alpha$) ;
4     if $|\mathcal{C}| < B$ then
5         initialize a new counter for $\mathcal{I}_s(\alpha)$
      else
6         abort current process
7 Return $(n_{\max})$
```

Using these processes, we design our algorithm as follows. Let ℓ be an integer to be chosen later. Choose $k = \lfloor \epsilon n \rfloor^{1/\ell}$ and $B = \lceil k/\epsilon \rceil$. For $i = 0$ to ℓ, in parallel, run the (k^i, B)-process. At the end of the stream, output $n_{\max}$ corresponding to the smallest i such that the (k^i, B)-process did not abort. Let n^* be this output. There will always exist a suitable i because, as can be checked easily, the (k^ℓ, B)-process cannot abort.

▶ **Claim 2.2.** *We have* $(1-\epsilon)OPT \leqslant n^* \leqslant OPT$.

Proof. The upper bound on n^* follows directly from the observation we recorded. For the lower bound, first suppose that the $(1, B)$-process did not abort. Then that process accurately counted $\mathcal{I}_s(\alpha)$ for *every* s in the stream, so $n^* = OPT$. Next, suppose that the (k^{i-1}, B)-process aborted, where $i > 0$. By our observation, $OPT > k^{i-1}B$. Also, because the (k^i, B)-process did not abort, by the other part of our observation, we have $n^* \geqslant OPT - k^i = OPT - (k^{i-1}B)(k/B) > (1 - k/B)OPT \geqslant (1-\epsilon)OPT$. ◀

The previous algorithm uses at most B counters and one extra register in each of its $\ell + 1$ parallel processes. Therefore, its space usage is $O(\ell B) = O(\ell \epsilon^{-1}(\epsilon n)^{1/\ell})$ words. We optimize this by setting $\ell \approx \log(\epsilon n)$.

▶ **Theorem 2.3.** *For all* $\epsilon \in (0,1)$*, there is a deterministic* $(1-\epsilon)$*-factor approximation for* MAX-DENSE-SORTED*, using* $O(\epsilon^{-1}\log(\epsilon n))$ *words of space.*

2.2 Points in Arbitrary Order: The Problem max-dense

We return to MAX-DENSE, with points arriving in an arbitrary (unsorted) order. Here, we assume that no two points in $\mathcal{S}$ have the same x-coordinate. For some appropriate constant c, we sample independently every element from the stream with probability $p = cn_k \log n/n$, where n_k is the size of a k-sample (for some appropriate k depending on the application). Let R' denote this sample. We sort the elements in R' and then choose a subset $R \subset R'$ by selecting every $c \log n$-th element from the sorted sequence of R'.

▶ **Claim 2.4.** *For a given* ϵ $(0 < \epsilon < 1)$*, there exists some appropriate* c *such that for every pair of consecutive elements* $r_i, r_{i+1} \in R$*, we have*

$$\Pr[k(1-\epsilon) \leqslant |\mathcal{S} \cap [r_i, r_{i+1}]| \leqslant k(1+\epsilon)] \geqslant 1 - 1/n.$$

Proof. For any consecutive (sorted) sample points r_i, r_{i+1}, if the number of unsampled elements, U_i is less than $k(1-\epsilon)$ elements, it implies that more than $c \log n$ elements were chosen from U_i. Every element is sampled independently with probability $p = \frac{c \log n}{k}$

($n_k = n/k$), so the expected number of samples in U_i is $c(1-\epsilon)\log n$. Let X_i be a random variable representing the number of samples from U_i. From Chernoff bounds, we have $\Pr[X_i \geqslant (1+\delta)\mathbb{E}[X_i]) \leqslant \exp(-\delta^2\mathbb{E}[U_i]/2)$ where $\mathbb{E}[U_i] = c(1-\epsilon)\log n$ and $1+\delta = 1/(1-\epsilon)$, i.e., $\delta \approx \epsilon$. For an appropriately large value of $c = \Omega(\frac{1}{\epsilon^2})$, we can bound this probability by $\frac{1}{n^2}$. The above calculation holds for a pair of sample points that are consecutive, but we can easily uncondition it by multiplying with the probability that they are consecutive (which is less than 1). Therefore, none of the intervals contain less than $k(1-\epsilon)$ points.

A similar calculation yields an upper bound on the number of unsampled elements in an interval. ◀

The above proof says that the sample R can be treated as a k-sample of $\mathcal{S}$ for getting an approximate solution for MAX-DENSE. If $OPT \geqslant \Delta$, then we choose $k = \lfloor \epsilon\Delta \rfloor$ to have the size of the k-sample as $n_k = n/(\epsilon\Delta)$. If OPT_R is the maximum count of an α-interval corresponding to an element of R, then with high probability we have $(1-\epsilon)^2 OPT \leqslant k(OPT_R - 1) \leqslant (1+\epsilon)^2 OPT$, where the extra $1 \pm \epsilon$ factor is due to the sampling variance as per the previous claim. Substituting $\epsilon = \epsilon/2$, we have an $(1-\epsilon)$ approximation.

▶ **Theorem 2.5.** *If $OPT \geqslant \Delta$, then for any $0 < \epsilon < 1$, OPT can be approximated within a factor $(1-\epsilon)$ with high probability using $O\left((\epsilon\Delta)^{-1} cn \log n\right)$ space where $c = O(\epsilon^{-2})$. The value Δ is an input to the streaming algorithm.*

▶ Remark. There is a huge gap between the space requirements of MAX-DENSE and MAX-DENSE-SORTED.

An output sensitive algorithm

With the strong lower bound already shown in Theorem 2.1, our goal is to have an output sensitive algorithm. Here, the intervals (of unequal length) are created online, and stored in a height balanced binary tree $\mathcal{T}$. We use the term *short*, *exact* and *long* to denote the intervals having length less than or equal to or greater than α. With each created interval I, we store its *span* $\delta(I)$ and *count* fields $count(I)$. Every interval J having span $\delta(J) > \alpha$ has $count(J) = 0$. Initially, a single interval $(-\infty, \infty)$ is present in $\mathcal{T}$. When a point p arrives, the tree $\mathcal{T}$ is searched to identify the interval (say $I = [a, b]$) containing p. If $\delta(I) \leqslant \alpha$, $count(I)$ is incremented. If $\delta(I) > \alpha$, we insert an interval J with one end point at p and of span $\delta(J) = \alpha$ in $\mathcal{T}$. Here the following cases need to be considered:

Case 1: $J = [p, q]$ is contained in an existing interval $I = [a, b]$ with $\delta(I) > \alpha$ (see Figure 1(a)). We delete I from $\mathcal{T}$ and insert three intervals $I_1 = [a, p]$, $I_2 = J = [p, q]$ and $I_3 = [q, b]$ in $\mathcal{T}$ with $count(I_1) = count(I_3) = 0$ and $count(I_2) = 1$. Here I_2 is an *exact* interval. I_1 and I_3 may be of any type.

Case 2: If $J = [p, q]$ overlaps with the some other interval $I' = [c, d]$ ($\neq I$) then I' must be an α-interval. We consider $J' = [q', p]$ of length α, with p as its right end-point (where q' is not an input point).

Case 2.1: If J' does not overlap with any other interval (See Figure 1(b)), then we delete I and insert three intervals $I_1 = [a, q], I_2 = J' = [q, p], I_3 = [p, b]$ in $\mathcal{T}$. Here I_2 is *exact* and I_3 is *short*. I_1 may be of any type.

Case 2.2: If $J' = [q, p]$ overlaps with an interval $I'' = [e, f]$ (See Figure 1(c)), then I'' is also of length α, and we have $I = [a, b] = [f, c]$ with $\alpha < \delta(I) \leqslant 2\alpha$. Here I is replaced with an *exact* interval $I_1 = [a, a+\alpha]$ and a *short* interval $I_2 = [a+\alpha, b]$ in $\mathcal{T}$ with $count[I_1] = 1$ and $count[I_2] = 0$.

The intervals created are characterized as follows.

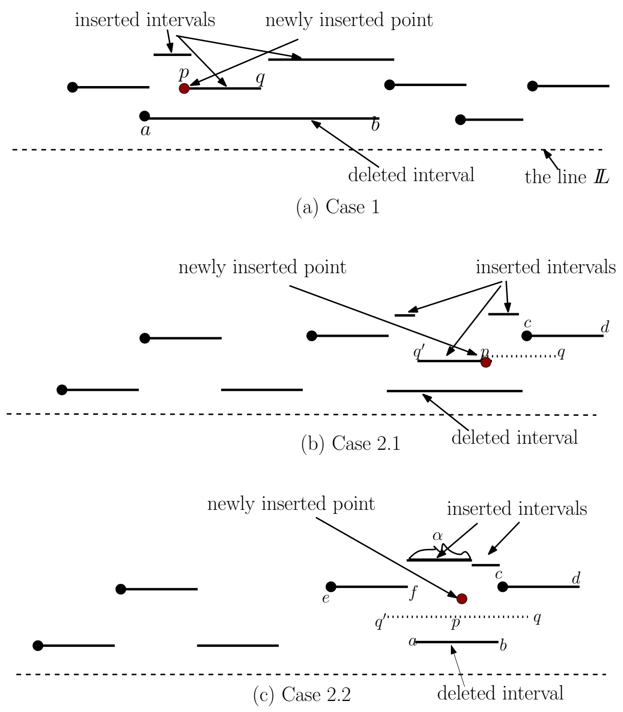

Figure 1 Processing of a new point in the stream: Here the dotted line is $\mathbb{L}$, the existing intervals, and the intervals to be inserted for a new stream element p, are shown.

▶ **Lemma 2.6.** *(a) At any point of time during the execution, the two adjacent intervals of any* short *interval are* exact *intervals.*

(b) The interval with maximum frequency contains at least $\frac{1}{3}OPT$, where OPT is the maximum number of points of S that an α-interval can contain.

Proof. Part (a) follows from the fact that a *short* interval is created by splitting a long interval of length less than 2α (see Case 2.2 earlier).

Part (b) follows from the fact that the interval corresponding to the OPT may span at most 3 intervals of $\mathcal{T}$. ◀

We now give an estimate of the size of the work-space for maintaining $\mathcal{T}$ using the size of maximum independent set (MIS) of the set of α-intervals anchored at each point of $\mathcal{S}$.

▶ **Lemma 2.7.** *If χ is the number of* exact *intervals in $\mathcal{T}$, then $\frac{2}{3}MIS \leqslant \chi \leqslant MIS$.*

Proof. The right-hand side of the inequality trivially follows. We need to prove the left-hand side of the inequality. Note that, the *long* intervals do not contain any point. Also, in the optimum solution, it is not possible to have more than one interval generated by two points inside a *short* or *exact* interval since its length is less than or equal to α. Again, both the neighbors of a *short* interval in $\mathcal{T}$ are *exact* intervals (by Lemma 2.6(a)). Thus, we have the left hand side of the inequality, since in the worst case, there may exist an instance where each triplet (*exact, short, exact*) of intervals is separated by a *long* interval. ◀

The output sensitive algorithm is summed up in the next Theorem. Note that MIS in the worst case can be linear.

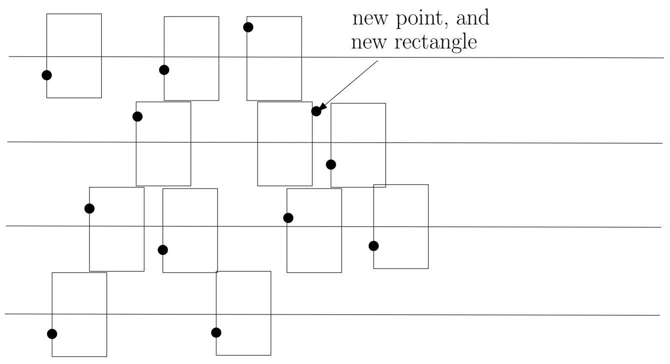

Figure 2 Insertion of a point.

▶ **Theorem 2.8.** *For* MAX-DENSE*, an α-interval $\hat{I}$ can be computed in $O(n \log n)$ time using $O(MIS)$ extra work-space, such that the number of points of S that $\hat{I}$ covers is at least $\frac{1}{3}OPT$, where OPT is the maximum number of points of S that can lie inside an α-interval, and MIS is the size of the independent set among all α-intervals with left end-points anchored at the points in S.*

Proof. The approximation factor follows from Lemma 2.6(b). The time complexity follows from the fact that we are spending $O(1)$ time for processing each point. The space complexity follows from the number of intervals stored in $\mathcal{T}$. By Lemma 2.6(a), the number of short intervals is less than the number of *exact* intervals. The number of *large* intervals can be at most 1 more than the number of *exact* intervals in the worst case, and Lemma 2.7 says that the number of exact intervals is less than or equal to MIS. Thus, the number of intervals in $\mathcal{T}$ is $O(MIS)$. ◀

2.3 Maximum Density with Points in Two Dimensions

Here a stream of points $\mathcal{S}$ is arriving online in $\mathbb{R}^2$, and a rectangular window W of size $\alpha \times \beta$ is given. The objective is to report the position of W that contains maximum number of points of $\mathcal{S}$. We can formulate the problem as follows.

▶ **Definition 2.9.** For each point p in the stream $\mathcal{S}$, an *exact copy of* W is a rectangle of size $\alpha \times \beta$ with p on its top-left corner.

Thus, our objective is to compute the largest clique in the intersection graph of these exact copies of W, where the bottom-right corner of W is to be placed to contain the maximum number of points. In order to handle this streaming version of the problem, we create copies of W in a slightly different manner, and show that the approximation factor of our proposed algorithm is 6.

As in MAX-DENSE, here also we create a covering of points of $\mathcal{S}$ in $\mathbb{R}^2$ with disjoint rectangles of size $\alpha' \times \beta$, $\alpha' \leqslant \alpha$, such that each rectangle contains at least one point of $\mathcal{S}$. As the points in $\mathcal{S}$ arrive, we create these rectangles online (see Figure 2), and store them in a data structure. When a point $p \in \mathcal{S}$ arrives, if it lies inside an existing rectangle then the *count* of that rectangle is increased by one; otherwise, a new rectangle is created that contains p, and its count is set to 1. When the stream ends, the rectangle having the maximum *count* is reported.

We assume that the points in $\mathcal{S}$ have positive x and y coordinates. We conceptually split the floor using horizontal lines $y = 0, \beta, 2\beta, \ldots$. On arrival of a point $p = (p_x, p_y) \in \mathcal{S}$, if it is not inside any one of the existing rectangles, we create a new rectangle as follows: we compute $i = \frac{p_y}{\beta}$. The vertical span of the created rectangle is $[i\beta - \frac{\beta}{2}, i\beta + \frac{\beta}{2}]$ or $[(i+1)\beta - \frac{\beta}{2}, (i+1)\beta + \frac{\beta}{2}]$ depending on whether $p_y - i\beta < \frac{\beta}{2}$ or $p_y - i\beta > \frac{\beta}{2}$. As in MAX-DENSE, the horizontal span of this rectangle is decided such that it must contain p, its horizontal width is at most α, and it does not overlap on any other existing rectangles in the data structure.

We store the horizontal lines containing at least one rectangle in a height-balanced binary tree $\mathbb{T}$. The rectangles having vertical span $[i\beta - \frac{\beta}{2}, i\beta + \frac{\beta}{2}]$ are stored in the form of disjoint intervals on the horizontal line $y = i\beta$ in a height-balanced binary tree $\mathcal{T}_i$ as in MAX-DENSE, and is attached with the i-th node of $\mathbb{T}$. The following theorem generalizes the result.

▶ **Theorem 2.10.** *Given a stream $\mathcal{S}$ of n points in $\mathbb{R}^2$, executing a single pass over the stream, one can compute a position of placing a rectangular window W of a given size in $\mathbb{R}^2$ such that it encloses at least $\frac{OPT}{6}$ points, where OPT is the maximum number of points in $\mathcal{S}$ that can be enclosed by placing the window W.*

The time and work-space required for executing this algorithm is $O(n \log R_{opt})$ and $O(R_{opt})$ respectively, where R_{opt} is the size of the maximum independent set of the exact copies *of W corresponding to the points in $\mathcal{S}$.*

Proof. Let W_{opt} be the optimum position of the rectangle W, and OPT be the number of points in W_{opt}. Our algorithm has reported W_{max} that contains maximum number of points with respect to our definition of rectangles for covering the points in the plane. Observe that W_{opt} can overlap on at most 6 different rectangles according to our layout (see Figure 2); one of these rectangles must contain at least $\frac{1}{6}OPT$ number of points. Thus if OPT be the number of points in W_{opt}, then W_{max} contains at least $\frac{1}{6}OPT$ points.

If M is the number of rectangles present in the data structure $\mathbb{T}$, then, processing each point takes $O(\log M)$ time in the worst case. Thus, the time complexity of the algorithm is $O(n \log M)$ time, and it uses $O(M)$ extra work-space.

Now, we show that $M \leqslant 2R_{opt}$. Consider the *exact copy of* W corresponding to a point $p \in \mathcal{S}$ (see Definition 2.9). If $i = \lfloor \frac{p_y}{\beta} \rfloor$, then assign p (and hence, W) to both the lines $y = i\beta$ and $y = (i+1)\beta$. Now, consider each horizontal line separately, and consider the intersection graph of the intervals of width α corresponding to the assigned points with this line. If I_i is the maximum independent set of this interval graph, and M_i is the set of intervals stored in $\mathcal{T}_i$, then $|M_i| \leqslant |I_i|$ (by Lemma 2.7). Thus, $M = \sum_{i=1}^{k} |M_i| \leqslant \sum_{i=1}^{k} |I_i|$, where k is the number of horizontal splitting lines of the floor. Again, since the exact copy of W corresponding to each point $p \in \mathcal{S}$ is assigned to two adjacent splitting lines, the two sets $I_{odd} = \cup_{i=1,3,\ldots,k} I_i$ and $I_{even} = \cup_{i=2,4,\ldots,k} I_i$ are independent, and the size of each of them is less than or equal to R_{opt}. Thus, we have the desired result $M = \sum_{i=1}^{k} |M_i| \leqslant \sum_{i=1}^{k} |I_i| \leqslant I_{odd} + I_{even} \leqslant 2R_{opt}$. ◀

3 Threshold and Emptiness Queries

Now we turn to the other types of interval queries, namely threshold and emptiness queries respectively.

3.1 Threshold Queries: The Problem threshold

Recall that the goal of THRESHOLD is that given prespecified α and Δ, to determine whether an α-interval can be placed to contain at least Δ of the points in the input stream $\mathcal{S}$. As already noted, this is equivalent to finding whether there exists an $s \in \mathcal{S}$ such that $|\mathcal{I}_s(\alpha)| \geqslant \Delta$. We first discuss a two-pass deterministic algorithm for THRESHOLD, followed by a one-pass randomized approximation algorithm.

Algorithm 2: Update(s)

```
begin
    Initialize all $\lceil 1/\epsilon \rceil$ counters to 0;
    if (s belongs to any of the intervals being tracked by counters in $\mathcal{C}$) then
        increment the counter for the $\alpha$-interval in $\mathcal{L}$ in which $s$ belongs;
    else
        if ($|\mathcal{C}| < \lceil 1/\epsilon \rceil$) then
            Open a new counter, that tracks the number of points inside $\mathcal{I}_i(\alpha)$, with a
              count of 1, where $i = \lceil s/\alpha \rceil$;
        else
            Decrement all counters in $\mathcal{C}$ by 1 and return to the available pool of
              counters all counters that reach 0;
```

A Deterministic Algorithm

The idea is to use the Misra-Gries summary [21] which is basically a generalization of the classical majority finding algorithm. Subsequent researchers [12, 18, 25] have used this idea for frequency estimation and finding heavy hitters. Apart from the sequence $\mathcal{S}$, we have a threshold ϵ, $0 < \epsilon < 1$, and we can maintain an estimate $\bar{f}_i$, of $f_i = |I_{s_i}(\alpha)|$, which is the frequency of $s_i \in \mathcal{S}$, such that $f_i - \epsilon n \leqslant \bar{f}_i \leqslant f_i$, and $\bar{f}_i$ for all α-intervals can be computed using $O(1/\epsilon)$ counters.

We reduce our problem to Misra-Gries summary giving labels to the points in $\mathcal{S}$. Each point $s \in \mathcal{S}$ is labeled as $\lceil s/\alpha \rceil$ – this basically classifies each point s into a set of disjoint canonical intervals $\mathcal{L} = \{\mathcal{I}_1(\alpha) = [0, \alpha), \mathcal{I}_2(\alpha) = [\alpha, 2\alpha), \ldots, \}$. Let us denote the set of counters as $\mathcal{C}$, $|\mathcal{C}| \leqslant \lceil 1/\epsilon \rceil$; the counters in $\mathcal{C}$ would maintain the count of points in some of the $\lceil 1/\epsilon \rceil$ intervals of $\mathcal{L}$. We set $\Delta = \epsilon \cdot n$. Note that, at a time at most $\lceil 1/\epsilon \rceil$ α-intervals are active. The procedure is described next.

Let $f_i = |\mathcal{I}_i(\alpha)|$, denote the number of points inside the i-th canonical interval of $\mathcal{L}$. At the end of the stream, if $\bar{f}_i$ be the value of the counter for the i-th canonical interval of $\mathcal{L}$, ($\bar{f}_i = 0$ if it is decremented to 0 during the process), then it is guaranteed that $\bar{f}_i \in [f_i - \Delta, f_i]$ because of the following ideas as described in [25]. The upper bound is trivial. For the lower bound, let $\bar{f}_i \geqslant f_i - \gamma$ where γ is the number of times the counter for an α-interval can be decremented. Recall that $|\mathcal{C}|$ counters are decremented together. As all α-intervals are disjoint and no point is repeated, we have $\lceil 1/\epsilon \rceil \cdot \gamma \leqslant n$. So, $\gamma \leqslant n\epsilon \leqslant \Delta$. Thus we have $f_i - \Delta n \leqslant \bar{f}_i \leqslant f_i$. Note that, the converse is not true, i.e. even if $\bar{f}_i > 0$ for some s_i, f_i may be less than Δ. This necessitates a second pass, where we can verify the actual counts.

The above gives information only about the canonical intervals. Now using the ideas of the simple 2-approximation algorithm in Section 2.1, we can claim the following about the original question of THRESHOLD– if there exists an s with $|\mathcal{I}_s(\alpha)| \geqslant \Delta$, then there also exists a canonical interval with frequency greater than $\Delta/2$. So, if $\bar{f}_i \geqslant \Delta$, then our answer is yes; if all $\bar{f}_i \leqslant \Delta/2$, then our answer is no. If there exists $\bar{f}_i$ such that $\Delta/2 \leqslant \bar{f}_i \leqslant \Delta$, then the only thing we can say about the THRESHOLD question is that there exists s with $|\mathcal{I}_s(\alpha)| \geqslant \Delta/2$.

▶ **Theorem 3.1.** *There exists a two-pass deterministic algorithm for* THRESHOLD *using $O(1/\epsilon)$ counters that gives a 2-factor approximate answer, where $\Delta = \epsilon \cdot n$.*

A Randomized Approximation Algorithm

We propose a one pass randomized approximation algorithm for THRESHOLD that returns the correct answer with high probability $(1 - n^{-\Omega(1)})$. We draw a random sample of points from S where each point is chosen with probability $p = \frac{\log n}{\Delta}$ to generate a random sample R. But n is unknown to us since it is an one pass algorithm. Assume, for now that we know n; we would later resolve this problem.

Note that, the expected space needed for storing R is $\frac{n \log n}{\Delta}$. As every element in R is sampled with a probability p, the expected sample size in any α-interval I containing more than Δ points is $R_I = \Omega(\log n)$. Using Chernoff bounds, it can be shown that with high probability $R_I \geqslant c \log n$. Moreover, if all α-intervals contain fewer than $\frac{\Delta}{\beta}$ points, for some $\beta > 1$, then with high probability, no α-interval contains more than $(c - \epsilon) \log n$ points, for some $\epsilon > 0$. If the given space exceeds $\frac{n \log n}{\Delta}$, then the above scheme works in a straight forward manner by first choosing the sample and subsequently finding the largest α-interval of the sample and then verifying if it exceeds $c \log n$. To extend this idea where n is not known a priori, we can use the idea of Manku and Motwani [20] where the sampling rate is decreased as the stream progresses, so that space usage remains bounded. We can summarize as follows.

▶ **Theorem 3.2.** *There is a one pass $O(\frac{n \log n}{\epsilon^2 \cdot \Delta})$ space bounded randomized algorithm that correctly reports an α-interval containing more than Δ points or asserts that no α-interval contains more than $(1 - \epsilon) \cdot \Delta$ points with high probability.*

▶ Remark. Compared to the two-pass algorithm, it uses $\log n$-factor more space. Compared to Theorem 2.5, the bound is better by a factor ϵ since we are only interested in a threshold.

Space Lower Bound for Threshold Queries

We can obtain a lower bound for THRESHOLD by using the same technique as for MAX-DENSE, i.e., reducing from the F_∞ problem. Suppose a stream of integers $\mathcal{S}$ has the property that either $F_\infty(\mathcal{S}) = 1$ or else $F_\infty(\mathcal{S}) \geqslant \Delta$, for some threshold parameter Δ. Taking $\alpha = 0$ we see that the answers to the threshold query in these two cases are "no" and "yes" respectively. We conclude the following lower bound. As before, the implicit hard instances can be made non-degenerate by perturbation.

▶ **Theorem 3.3.** *A randomized constant-pass algorithm that solves the basic decision version of the* THRESHOLD *problem with parameter Δ requires $\Omega(n/\Delta^2)$ space.*

3.2 Emptiness Queries: The Problem emptiness

In the EMPTINESS problem, the objective is to find whether there exists an empty interval of length α within the extent of the set of points in the data stream. As noted earlier, this is equivalent to determining whether there exists $s \in \mathcal{S} \setminus \{\max \mathcal{S}\}$ such that $|\mathcal{I}_s(\alpha)| = 1$. We show that this problem also admits strong lower bounds.

For this, we reduce from DISJ$_m$, the two-party SET-DISJOINTNESS communication problem on the universe $\mathcal{M} = \{1, 2, \ldots, m\}$. In the communication problem, Alice gets a set $X \subseteq \mathcal{M}$ and Bob gets a set $Y \subseteq \mathcal{M}$. They must decide whether or not $X \cap Y = \varnothing$. This problem has deterministic communication complexity $m + 1$ and randomized communication complexity $\Omega(m)$, see e.g., [24].

The reduction is as follows. Alice converts X into a stream of elements of $\{0\} \cup (\mathcal{M} \setminus X)$ and Bob similarly converts Y to $(\mathcal{M} \setminus Y) \cup \{m + 1\}$. If $X \cap Y = \varnothing$, then the concatenation of these streams contains every point in $\{0, 1, \ldots, m+1\}$, so it is impossible to find an empty interval of width $\alpha = \frac{3}{2}$. On the other hand, if X and Y contain a common element z, then the combined stream is missing z, so the open interval $(z-1, z+1)$ is empty. It follows that any algorithm that solves EMPTINESS also solve DISJ_m. The stream created has length $n \leqslant 2m$. Therefore, we obtain the following bounds.

▶ **Theorem 3.4.** *Every randomized constant-pass algorithm that solves* EMPTINESS *requires* $\Omega(n)$ *bits of space. Furthermore, every deterministic algorithm that does the same requires at least* $\frac{1}{2}n - O(1)$ *space.*

4 Conclusion

We studied some problems related to density of points inside intervals in the streaming model. We observed that these problems in geometry are generalizations of frequency moments, frequency estimation and heavy hitters problems. We obtained deterministic as well as randomized approximations using bounded amount of extra space. We proved nearly matching lower bounds on space as well. An interesting open problem would be to look at the higher dimensional variants of the above problems apart from tightening the space bounds.

References

1 Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, July 2004.

2 A. Aggarwal and S. Suri. Fast algorithms for computing the largest empty rectangle. In *Proceedings of the Third Annual Symposium on Computational Geometry*, SCG'87, pages 278–290, 1987.

3 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

4 T. Asano, M. Sato, and T. Ohtsuki. Computational geometric algorithms. In *Layout Design and Verification, Advances in CAD for VLSL (Edited by T. Ohtsuki)*, pages 295–347. North Holland, 1986.

5 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, 3rd ed. edition, 2008.

6 Sergio Cabello and Pablo Pérez-Lantero. Interval selection in the streaming model. In *WADS'15*, pages 127–139, 2015.

7 Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *PROC18 # CCC*, pages 107–117, 2003.

8 Timothy M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom.*, 35(1-2):20–35, 2006.

9 Timothy M. Chan and Vinayak Pathak. Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. In *Proceedings of the 12th International Conference on Algorithms and Data Structures*, WADS'11, pages 195–206, 2011.

10 Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

11 Graham Cormode and Marios Hadjieleftheriou. Methods for finding frequent items in data streams. *VLDB J.*, 19(1):3–20, 2010.

12 Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Frequency estimation of internet packet streams with limited space. In *Algorithms – ESA 2002, 10th Annual European Symposium*, pages 348–360, 2002.
13 Yuval Emek, Magnús M. Halldórsson, and Adi Rosén. Space-constrained interval selection. In *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 302–313, 2012.
14 Sariel Har-Peled and Soham Mazumdar. Fast algorithms for computing the smallest k-enclosing circle. *Algorithmica*, 41(3):147–157, 2005.
15 Monika Rauch Henzinger, Prabhakar Raghavan, and Sridar Rajagopalan. Computing on data streams, 1998.
16 John Hershberger and Subhash Suri. Adaptive sampling for geometric problems over data streams. *Comput. Geom. Theory Appl.*, 39(3):191–208, April 2008.
17 Piotr Indyk and David P. Woodruff. Optimal approximations of the frequency moments of data streams. In *STOC*, pages 202–208, 2005.
18 Richard M. Karp, Scott Shenker, and Christos H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. Database Syst.*, 28:51–55, 2003.
19 Subhashis Majumder and Bhargab B. Bhattacharya. On the density and discrepancy of a 2d point set with applications to thermal analysis of vlsi chips. *Inf. Process. Lett.*, 107(5):177–182, 2008.
20 Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *VLDB*, pages 346–357, 2002.
21 Jayadev Misra and David Gries. Finding repeated elements. *Sci. Comput. Program.*, 2(2):143–152, 1982.
22 S. Muthukrishnan. Data streams: Algorithms and applications. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'03, pages 413–413, 2003.
23 Subhas C. Nandy and Bhargab B. Bhattacharya. A unified algorithm for finding maximum and minimum point enclosing rectangles and cuboids. *Int. J. on Computers and Mathematics with applications*, 29(8):45–61, 1995.
24 Alexander Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.
25 Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. Parallel streaming frequency-based aggregates. In *26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA'14*, pages 236–245, 2014.
26 David P. Woodruff. Frequency moments. In *Encyclopedia of Database Systems*, pages 1169–1170. Springer, US, 2009.

A Maximum number of points in an interval having uniformly distributed points

Let the stream $\mathcal{S}$ be any arbitrary permutation of n points where the points are uniformly distributed in the range $[\ell, u]$. Let $\frac{u-\ell}{\alpha} = g(n)$ in where $g(n)$ is a function of n. Then we have the following bound on the maximum number of points in any interval of length α.

▶ **Lemma 1.1.** *The maximum number of points in any interval $I \subset [\ell, u]$ is bounded by $\max\{\frac{n}{g(n)}, c \log n / \log\log n\}$ with probability at least $1 - 1/n$ for $g(n) \leqslant n^{1+\epsilon}$ for any $\epsilon > 0$. For $g(n) \geqslant n^{1+\epsilon}$, it is $O(1)$ with probability $\geqslant 1 - 1/n$.*

Proof. Consider a canonical partition of the range $[\ell, u]$ consisting of intervals $[\ell, \ell + \alpha], [\ell + \alpha, \ell + 2\alpha] \ldots [\ell + i\alpha, \ell + (i+1)\alpha]$. Let us denote this set of intervals by $\mathcal{C}$ - from our previous assumption, the number of intervals in $\mathcal{C}$ is bounded by $g(n)$.

Suppose the n points are generated as i.i.d. in $[\ell, u]$, viz., each of the n points is independently generated with uniform distribution in $[\ell, u]$. For a fixed interval $I' \in \mathcal{C}$, the probability that point $q_i, 1 \leqslant i \leqslant n$ is in I' is $p = \frac{\alpha}{u-\ell}$. Let U be random variable that represents the number of points in I' that follows a binomial distribution, so $\mathbb{E}[U] = \frac{n}{g(n)}$. Therefore it follows from the following version of Chernoff bounds

$$\Pr[U \geqslant (1+\Delta)\mathbb{E}[U] \leqslant \left[\frac{e^{\Delta}}{(1+\Delta)^{1+\Delta}}\right]^{E[U]} \tag{1}$$

that the number of points in I' is bounded by $c' \log g(n)/\log\log g(n)$ with probability 1 - $1/g(n)^{c'}$ for some appropriate c' using $\Delta = \frac{\ell-u}{\alpha}$, for $g(n) \geqslant en$. Since the total number of intervals is bounded by $g(n)$, a similar bound follows for all intervals in $\mathcal{C}$ using the union bound and by adjusting the value of c'. For any arbitrary α length interval ($\notin \mathcal{C}$), it intersects at most two intervals in $\mathcal{C}$ and so it cannot exceed $2c \log g(n)/\log\log g(n) = \theta(\frac{\log n}{\log\log n})$.

For $g(n) \leqslant n/\log n)$, the bound of $n/g(n)$ holds with high probability using similar calculations.

For $g(n) \geqslant n^{1+\epsilon}$, $E[U] = n^{-\epsilon}$ and choose $\Delta = cn^{\epsilon}$ for some appropriately large constant c. Substituting in Equation 1 yields the required bound $\Pr[U \geqslant \Omega(1)] \leqslant 1/n$. ◀