Weighted Maximum Independent Set of Geometric Objects in Turnstile Streams

Ainesh Bakshi
Carnegie Mellon University, Pittsburgh, PA, USA
abakshi@cs.cmu.edu

Nadiia Chepurko
MIT, Cambridge, MA, USA
nadiia@mit.edu

David P. Woodruff
Carnegie Mellon University, Pittsburgh, PA, USA
dwoodruf@cs.cmu.edu

Abstract

We study the Maximum Independent Set problem for geometric objects given in the data stream model. A set of geometric objects is said to be independent if the objects are pairwise disjoint. We consider geometric objects in one and two dimensions, i.e., intervals and disks. Let α be the cardinality of the largest independent set. Our goal is to estimate α in a small amount of space, given that the input is received as a one-pass stream. We also consider a generalization of this problem by assigning weights to each object and estimating β, the largest value of a weighted independent set. We initialize the study of this problem in the turnstile streaming model (insertions and deletions) and provide the first algorithms for estimating α and β.

For unit-length intervals, we obtain a $(2+\epsilon)$-approximation to α and β in poly$(\frac{\log(n)}{\epsilon})$ space. We also show a matching lower bound. Combined with the 3/2-approximation for insertion-only streams by Cabello and Perez-Lanterno [11], our result implies a separation between the insertion-only and turnstile model. For unit-radius disks, we obtain a $\left(\frac{8\sqrt{3}}{\pi}\right)$-approximation to α and β in poly $\left(\frac{\log(n)}{\epsilon}\right)$ space, which is closely related to the hexagonal circle packing constant.

Finally, we provide algorithms for estimating α for arbitrary-length intervals under a bounded intersection assumption and study the parameterized space complexity of estimating α and β, where the parameter is the ratio of maximum to minimum interval length.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Streaming models

Keywords and phrases Weighted Maximum Independent Set, Geometric Graphs, Turnstile Streams

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.64

Category APPROX

Related Version `https://arxiv.org/abs/1902.10328`

Funding Ainesh Bakshi and David Woodruff acknowledge support in part from NSF No. CCF-1815840.

Acknowledgements Part of this work was done while Ainesh Bakshi and David Woodruff were visiting the Simons Institute for the Theorem of Computing.

1 Introduction

Maximum Independent Set (MIS) is a fundamental combinatorial problem and in general, is NP-Hard to approximate within a $n^{1-\epsilon}$ factor, for any constant $\epsilon > 0$ [34]. We focus on the MIS problem for geometric objects: we are given as input n intervals on the real line or disks in the plane and our goal is to output the largest set of non-overlapping intervals or disks. Computing the Maximum Independent Set of intervals and disks has numerous applications in scheduling, resource allocation, cellular networks, map labellings, clustering, wireless ad-hoc networks and coding theory, where it has been extensively studied [31] [4], [50], [12], [6], [5], [1], [33], [45].

In the one dimensional setting, the MIS problem, also known as the Interval Scheduling[1] problem, has a simple greedy algorithm that picks intervals in increasing order of their right endpoint to obtain an optimal solution. The variant with weighted intervals can also be solved in polynomial time using dynamic programming, which is shown in a number of modern algorithms textbooks [21], [41]. These algorithms have considerable applications in resource allocation and scheduling, where offline and online variants have been extensively studied and we refer the reader to [42] for a survey.

In the two dimensional setting, MIS of geometric objects, such as line segments [35], rectangles [30], [37] and disks [19], is *NP-Hard*. However, in the offline setting (polynomial space), a PTAS is known for fat objects (squares, disks) and pseudo-disks [15] (who also provide a recent survey). The MIS problem for arbitrary rectangles has also received considerable attention: [13] show a $\log\log(n)$ approximation in polynomial time and [18] obtain a $(1+\epsilon)$-approximation in $n^{\text{poly}(\log(n))\epsilon^{-1}}$ time for axis-aligned rectangles.

Streaming Model. The increase in modern computational power has led to massive amounts of available data. Therefore, it is unrealistic to assume that our data fits in RAM. Instead, working with the assumption that data can be efficiently accessed in a sequential manner has led to streaming algorithms for a number of problems. Several classical problems such as heavy-hitters and l_p sampling [38], l_p estimation [39], entropy estimation [44], [20], maximum matching [43] etc. have been studied in the turnstile model and recent work has led to interesting connections with linear sketches [2].

In this paper, we study the streaming complexity of the geometric MIS problem, where the input is a sequence of n updates, either inserting a new object or deleting a previously inserted object. We assume that the algorithm has poly-logarithmic bounded memory and at the end of the stream, the algorithm should output an estimate of the (weighted) cardinality of the MIS. Since most real world scheduling applications are dynamic, and scheduling constraints expire, it is crucial to allow for both insertions and deletions, while operating in the low-space setting. Consider the following concrete application: automatic point-label conflict resolution on interactive maps [48]. In this problem, the goal is to label features (geometric objects such as points, lines and polygons) on a map such that no two features with the same label overlap. Labelling maps in visual analytic software requires such labelling to be fast and dynamic, since features can be added and removed.

1.1 Our Contributions

We provide the first algorithmic and hardness results for the Weighted Maximum Independent Set (WMIS) problem for geometric objects in *turnstile streams* (where previously inserted objects may also be deleted). The aim of our work is to understand the MIS and WMIS problems in this common data stream model and we summarize the state of the art in Table 1. Our contributions are as follows:

1. **Unit-length Intervals.** Our main algorithmic contribution is a turnstile streaming algorithm achieving a $(2+\epsilon)$-approximation to α and β in $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space. We also show a matching lower bound, i.e., any (possibly randomized) algorithm approximating

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

Table 1 The best known upper and lower bounds for estimating α and β in insertion-only and turnstile streams (defined below). Note, the weight and length above are still polynomially bounded in n. The folklore result follows from partitioning the input into $O(\log(n))$ weight classes, estimating α on each one in parallel and taking the maximum estimate.

Problem	Insertion-Only Streams		Turnstile Streams	
	upper bound	lower bound	upper bound	lower bound
Unit Intervals Unit Weight	$3/2+\epsilon$ [10]	$3/2-\epsilon$ [10]	$2+\epsilon$ Thm 8	$2-\epsilon$ Thm 16
Unit Intervals Arbitrary Weight	$3/2+\epsilon$ Thm 22	$3/2-\epsilon$ [10]	$2+\epsilon$ Thm 8	$2-\epsilon$ Thm 16
Unit Disks Arbitrary Weight	$\frac{8\sqrt{3}}{\pi}+\epsilon$ Thm 21	$2-\epsilon$ Thm 25	$\frac{8\sqrt{3}}{\pi}+\epsilon$ Thm 21	$2-\epsilon$ Thm 16

α up to a $(2-\epsilon)$ factor requires $\Omega(n)$ space. Interestingly, this shows a strict separation between insertion-only and turnstile models since [10] show that a 3/2 approximation is tight in the insertion-only model.

An unintuitive yet crucial message here is that attaching polynomially bounded weights to intervals does not affect the approximation factor. Along the way, we also obtain new algorithms for estimating β in insertion-only streams which are presented in Section C.

2. **Arbitrary Length Intervals.** For arbitrary length intervals, we give a one-pass turnstile streaming algorithm that achieves a $(1+\epsilon)$-approximation to α under the assumption that the degree of the interval intersection graph is bounded by $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$. Our algorithm achieves $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space. We also study the problem for arbitrary lengths by parameterizing the ratio of the longest to the shortest interval. We give a one-pass turnstile streaming algorithm that achieves a $(2+\epsilon)$-approximation to α, where the space complexity is parameterized by W_{max}, which is an upper bound on the length of an interval assuming the minimum interval length is 1. Here, the space complexity of our algorithm is $\text{poly}\left(W_{max}\frac{\log(n)}{\epsilon}\right)$ and this algorithm gives sublinear space whenever W_{max} is sublinear.
3. **Unit-radius Disks.** We show that we can extend the ideas developed for unit-length intervals in turnstile streams to unit disks in the 2-d plane. We describe an algorithm achieving an $\left(\frac{8\sqrt{3}}{\pi}+\epsilon\right)$-approximation to α and β in $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space. One key idea in the algorithm is to use the hexagonal circle packing for the plane, where the fraction of area covered is $\frac{\pi}{\sqrt{12}}$ and our approximation constant turns out to be $4\cdot\frac{\sqrt{12}}{\pi}$.
We also show a lower bound that any (possibly randomized) algorithm approximating α or β for disks in insertion-only streams, up to a $(2-\epsilon)$ factor requires $\Omega(n)$ space. This shows a strict separation between estimating intervals and disks in insertion-only streams.

2 Related Work

There has been considerable work on streaming algorithms for graph problems. Well-studied problems include finding sparsifiers, identifying connectivity structure, building spanning trees, and matchings; see the survey by McGregor [46]. Recently, Cormode et. al. [22] provide guarantees for estimating the cardinality of a maximum independent set of general graphs via the Caro-Wei bound. Emek, Halldorsson and Rosen [24] studied estimating the cardinality of the maximum independent set for interval intersection graphs in insertion-only streams. They output an independent set that is a $\frac{3}{2}$-approximation to the optimal (OPT) for unit-length intervals and a 2-approximation for arbitrary-length intervals in $O(|\mathsf{OPT}|)$ space. Note that $|\mathsf{OPT}|$ could be $\Theta(n)$ which is a prohibitive amount of space.

Subsequently, Cabello and Perez-Lantero [10] studied the problem of estimating the *cardinality* of OPT, which we denote by α, for unit-length and arbitrary length intervals in one-pass insertion-only streams. For unit-length intervals in insertion-only streams, Cabello and Perez-Lantero [10] give a $(\frac{3}{2}+\epsilon)$ approximation to α in $\mathrm{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space. Additionally, they show that this approximation factor is tight, since any algorithm achieving a $(\frac{3}{2}-\epsilon)$-approximation to α requires $\Omega(n)$ space. For arbitrary-length intervals they give a $(2+\epsilon)$-approximation to α in $\mathrm{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space. Additionally, they show that the approximation factor is tight, since any algorithm achieving a $(2-\epsilon)$-approximation to α requires $\Omega(n)$ space. Recently, [23] studied MIS of intersection graphs in insertion-only streams. They show achieving a $(5/2-\epsilon)$-approximation to MIS of squares requires $\Omega(n)$ space.

To the best of our knowledge there is no prior work on the problem of Maximum Independent Set of unit disks in turnstile streams. In the offline setting, the first PTAS for MIS of disks was developed by [26] and later improved in running time by Chan [14], while [36] shows a PTAS for MIS of $k \times k$ squares. We note that these algorithms require space linear in the number of disks and use a dynamic programming approach that is not suitable for streaming scenarios.

We note that MIS can also be viewed as a natural generalization of the distinct elements problem that has received considerable attention in the streaming model. This problem was first studied in the seminal work of [29] and a long sequence of work has addressed its space complexity in both insertion-only and turnstile streams [3], [7], [32], [27], [28], [40], [9] and [23].

3 Notation and Problem Definitions

We let $D(d_j, r_j, w_j)$ be a disk in $\mathbb{R}^d$, where $d \in \{1, 2\}$, such that it is centered at a point $d_j \in \mathbb{R}^d$ with radius $r_j \in \mathbb{N}$ and weight w_j. We represent $D(d_j, r_j, w_j)$ using the short form D_j when d_j, r_j and w_j are clear from context. Note, we use the same notation to denote intervals in $d = 1$. For a set $\mathcal{P} \subseteq \mathbb{R}^d$ of n disks (unweighted or weighted), let G be the induced graph formed by assigning a vertex to each disk and adding an edge between two vertices if the corresponding disks intersect. We call G an intersection graph. The Maximum Independent Set (MIS) and Weighted Maximum Independent Set (WMIS) problems in the context of intersection graphs are defined as follows:

▶ **Definition 1** (Maximum Independent Set). *Let $\mathcal{P} = \{D_1, D_2 \ldots, D_n\} \subseteq \mathbb{R}$ be a set of n disks such that each weight $w_j = 1$ for $j \in [n]$. The MIS problem is to find the largest disjoint subset $\mathcal{S}$ of $\mathcal{P}$ (i.e., no two objects in $\mathcal{S}$ intersect). We denote the cardinality of this set by α.*

▶ **Definition 2** (Weighted Maximum Independent Set). *Let $\mathcal{P} = \{D_1, D_2 \ldots, D_n\} \subseteq \mathbb{R}^d$ be a set of n weighted disks. We let the weight $w_{\mathcal{S}}$ of a subset $\mathcal{S} \subseteq \mathcal{P}$ be $w_{\mathcal{S}} = \sum_{D_j \in \mathcal{S}} w_j$. The WMIS Problem is to find a disjoint (i.e., non overlapping) subset $\mathcal{S}$ of $\mathcal{P}$ whose weight $w_{\mathcal{S}}$ is maximum. We denote the weight of the WMIS by β.*

For a set $\mathcal{P}$ of disks, let $\mathsf{OPT}_{\mathcal{P}}$ denote MIS or WMIS of $\mathcal{P}$. We use $|\mathsf{OPT}_{\mathcal{P}}|$ to denote the cardinality of MIS as well as the weight of WMIS for $\mathcal{P}$. When the set $\mathcal{P}$ is clear from context, we omit it. Next, we define the two streaming models we consider. In our context, an *insertion-only* stream provides sequential access to the input, which is an ordered set of objects such that at any given time step a new interval arrives. *Turnstile streams* are an extension of this model such that at any time step, previously inserted objects can be deleted. An algorithm in the streaming model has access to space sublinear in the size of the input and is restricted to making one pass over the input.

For proving our lower bounds, we work in the two player one-way randomized communication complexity model, where the players are denoted by Alice and Bob, who have private randomness. The input of Alice is denoted by X and the input for Bob is denoted by Y. The objective is for Alice to communicate a message to Bob and compute a function $f : X \times Y \to \{0, 1\}$ on the joint inputs of the players. The communication is one-way and w.l.o.g. Alice sends one message to Bob and Bob outputs a bit denoting the answer to the communication problem. Let $\Pi(X, Y)$ be the random variable that denotes the transcript between sent from Alice to Bob when they execute a protocol Π.

A protocol Π is called a δ-error protocol for function f if there exists a function Π_{out} such that for every input $Pr[\Pi_{out}(\Pi(X, Y)) = f(X, Y)] \geq 1 - \delta$. The communication cost of a protocol, denoted by $|\Pi|$, is the maximum length of $\Pi(X, Y)$ over all possible inputs and random coin flips of the two players. The randomized communication complexity of a function f, $R_\delta(f)$, is the communication cost of the best δ-error protocol for computing f.

4 Technical Overview

In this section, we summarize our results and briefly describe the main technical ideas in our algorithms and lower bounds. We note that our results hold in the recently introduced Sketching Model [49]. This model captures applications of sketches in turnstile streams, distributed computing, communication complexity and property testing. While Sun et. al. study graph problems such as dynamic connectivity and triangle detection, we initiate the study of dynamic Maximum Independent Set in this model. While we state our results in for turnstile streams, they immediately extend to the sketching model.

4.1 Unit-length Intervals

Our main algorithmic contribution is to provide an estimate that obtains a $(2+\epsilon)$-approximation to WMIS of unit-length intervals in turnstile streams :

▶ **Theorem 3** (Theorem 8, informal). *For any $\epsilon > 0$, there exists a turnstile streaming algorithm that outputs an estimate such that with probability at least $99/100$, it is a $(2+\epsilon)$-approximation to WMIS of unit intervals (polynomially bounded weights) and the algorithm requires $\mathrm{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space.*

A naïve approximation. We start by describing a simple approach (Algorithm 1) to obtain a 9-approximation. The algorithm proceeds by imposing a grid of side length 1 and shifts it by a random integer. This is a standard technique used in geometric algorithms. We then snap each interval to the cell containing the center of the interval and partition the real line into odd and even cells. This partitions the input space such that intervals landing in distinct odd (even) cells are pairwise independent. Let $\mathcal{C}_e$ be the set of all even cells and $\mathcal{C}_o$ be the set of all odd cells.

By averaging, either $|\mathsf{OPT}_{\mathcal{C}_e}|$ or $|\mathsf{OPT}_{\mathcal{C}_o}|$ is at least $\frac{\mathsf{OPT}}{2}$, where $|\mathsf{OPT}|$ is the max weight independent set of intervals. We develop an estimator that gives a $(1+\epsilon)$-approximation to $|\mathsf{OPT}_{\mathcal{C}_e}|$ as well as $|\mathsf{OPT}_{\mathcal{C}_o}|$. Therefore, taking the max of the two estimators, we obtain a $(2+\epsilon)$-approximation to $|\mathsf{OPT}|$.

Having reduced the problem to estimating $|\mathsf{OPT}_{\mathcal{C}_e}|$, we observe that for each even cell only the max weight interval landing in the cell contributes to $\mathsf{OPT}_{\mathcal{C}_e}$. Then, partitioning the cells in $\mathcal{C}_e$ into poly$(\log(n))$ geometrically increasing weight classes based on the max weight interval in each cell and approximately counting the number of cells in each weight class suffices to estimate $|\mathsf{OPT}_{\mathcal{C}_e}|$ up to a $(1+\epsilon)$-factor.

Algorithm 1 Naïve Approximation.

Input: Given a turnstile stream $\mathcal{P}$ with weighted unit intervals, where the weights are polynomially bounded, ϵ and $\delta > 0$, Naïve Approximation outputs a $(9+\epsilon)$-approximation to β with probability $1-\delta$.

1. Randomly shift a grid Δ of side length 1. Partition the cells into even and odd, denoted by $\mathcal{C}_e$ and $\mathcal{C}_o$.
2. Consider a partition of cells in $\mathcal{C}_e$ into $b = \text{poly}(\log(n))$ weight classes $\mathcal{W}_i = \{c \in \mathcal{C}_e | (1+1/2)^i \leq m(c) < (1+1/2)^{i+1}\}$, where $m(c)$ is the maximum weight of an interval in c (this is not an algorithmic step since we do not know this partition a priori). Create a substream for each weight class $\mathcal{W}_i$ denoted by $\mathcal{W}'_i$.
3. For each new interval $D(d_j, 1, w_j)$, feed it to substream $\mathcal{W}'_i$ if $w_j \in [(1+1/2)^i, (1+1/2)^{i+1})$. For each substream $\mathcal{W}'_i$, maintain a $(1 \pm \epsilon)$-approximate ℓ_0-estimator (described below).
4. Let t_i be the ℓ_0 estimate corresponding to $\mathcal{W}'_i$. Let $X_e = \frac{2}{9(1+\epsilon)} \sum_{i \in [b]} (1+1/2)^{i+1} t_i$.
5. Repeat Steps 2-6 for the odd cells $\mathcal{C}_o$ to obtain the corresponding estimator X_o.

Output: $\max(X_e, X_o)$

Given such a partition, we can approximate the number of cells in each weight class by running an ℓ_0 norm estimator. Estimating the ℓ_0 norm of a vector in *turnstile streams* is a well studied problem and a result of Kane, Nelson and Woodruff [40] obtains a $(1 \pm \epsilon)$-approximation in poly$(\frac{\log(n)}{\epsilon})$ space. However, we do not know the partition of the cells into the weight classes a priori and this partition can vary drastically over the course of a stream given that intervals can be deleted. Therefore, the main technical challenge is to simulate this partition in *turnstile streams.*

As a first attempt, consider a partition of cells in $\mathcal{C}_e$ into $b = \text{poly}(\log(n))$ weight classes $\mathcal{W}_i = \{c \in \mathcal{C}_e | (1+1/2)^i \leq m(c) < (1+1/2)^{i+1}\}$, where $m(c)$ is the maximum weight of an interval in c. Create a substream for each weight class $\mathcal{W}_i$ and feed an input interval into this substream if its weight lies in the range $[(1+1/2)^i, (1+1/2)^{i+1})$. Let t_i be the corresponding ℓ_0 estimate for this substream. Approximate the contribution of $\mathcal{W}_i$ by $(1+1/2)^{i+1} \cdot t_i$. Sum up the estimates for all $i \in [b]$ to obtain an estimate for $|\mathsf{OPT}_{\mathcal{C}_e}|$.

We note that there are two issues with our algorithm. First, we overestimate the weight of intervals in class $\mathcal{W}_i$ by a factor of $3/2$ and second, for a given cell we sum up the weights of all intervals landing in it, instead of taking the maximum weight for the cell. In the worst case, we approximate the true weight of a contributing interval, $(3/2)^{i+1}$, with $\sum_{i'=1}^{i} (3/2)^{i'+1} \leq 3((3/2)^{i+1} - 1)$. Note, we again overestimate the weight, this time by a factor of 3. Combined with the approximation for the ℓ_0 norm, we obtain a weaker $(\frac{9}{2} + \epsilon)$-approximation to $|\mathsf{OPT}_{\mathcal{C}_e}|$ in the desired space. From our discussion above, this implies a $(9+\epsilon)$-approximation to $|\mathsf{OPT}|$. We also note that this attempt is not futile as we use the above algorithm as a subroutine subsequently.

A refined attempt. Next, we describe an algorithm that estimates $|\mathsf{OPT}_{\mathcal{C}_e}|$ up to a $(1+\epsilon)$-factor. Here, we use more sophisticated techniques to simulate a finer partition of the cells in $\mathcal{C}_e$ into geometrically increasing weight classes in turnstile streams. One key algorithmic tool we use here is a streaming algorithm for k-Sparse Recovery: given an input vector x such that x receives coordinate-wise updates in the turnstile streaming model and has at most k non-zero entries at the end of the stream of updates, there exist data structures that exactly recover x at the end of the stream. As mentioned in Berinde et al. [8], the k-tail guarantee is a sufficient condition for k-Sparse Recovery, since in a k-sparse vector, the elements of the tail are 0. We note that the Count-Sketch Algorithm [17] has a k-tail guarantee in *turnstile streams*.

This time around, we consider partitioning cells in $\mathcal{C}_e$ into poly $\left(\epsilon^{-1}\log(n)\right)$ weight classes, creating a substream for each one and computing the corresponding ℓ_0 norm. We also assume we know $|\mathsf{OPT}_{\mathcal{C}_e}|$ up to a constant (this can be simulated in *turnstile streams*). Formally, given $b = \text{poly}\left(\log(n), \epsilon^{-1}\right)$ weight classes, for all $i \in [b]$, let $\mathcal{W}_i$ denote the set of even cells with maximum weight sandwiched in the range $[(1+\epsilon)^i, (1+\epsilon)^{i+1})$. We then simulate sampling from the partition by subsampling cells in each $\mathcal{W}_i$ at the start of the stream, agnostic to the input. We do this at different sampling rates , i.e. for all $i \in [b]$, we subsample the cells in $\mathcal{W}_i$ with probability roughty $(1+\epsilon)^i/|\mathsf{OPT}_{\mathcal{C}_e}|$.

This presents several issues, as we cannot subsample non-empty cells in turnstile streams a priori. Further, if a weight class has a small number of non-empty cells, we cannot recover accurate estimates for the contribution of this weight class to $|\mathsf{OPT}_{\mathcal{C}_e}|$ at any level of the subsampling. To address the first issue, we agnostically sample cells from $\mathcal{C}_e$ according to a carefully chosen range of sampling rates and create a substream for each one. We then run a sparse recovery algorithm on the resulting substreams. At the right subsampling rate, we note that the resulting substream is sparse since we can filter out cells that belong to smaller weight classes. Further, we can ensure that the number of cells that survive from the relevant weight class (and larger classes) is small. Therefore, we recover all such cells using the sparse recovery algorithm.

To address the second issue, we threshold the weight classes that we consider in the algorithm based on the relative fraction of non-empty cells in them. This threshold can be computed in the streaming algorithm using the ℓ_0-norm estimates for each weight class. All the weight classes below the threshold together contribute at most an ϵ-fraction of $|\mathsf{OPT}_{\mathcal{C}_e}|$ and though we cannot achieve concentration for such weight classes, we show that we do not overestimate their contribution. Further, for all the weight classes above the threshold, we can show that sampling at the right rate can recover enough cells to achieve concentration.

We complement the above algorithmic result with a matching lower bound, i.e., a $(2-\epsilon)$-approximation to MIS, for any $\epsilon > 0$, requires $\Omega(n)$ space. This follows from an easy application of the Augmented Indexing problem. We note that our result combined with the 3/2-approximation by [11] implies an unexpected separation between insertion-only and turnstile streams.

4.2 Parametrized Algorithms for Arbitrary Length Intervals

In light of the lower bound discussed above, we identify two sources contributing to the streaming hardness of MIS for arbitrary length intervals : the number of pair-wise intersections (max-degree) and the ratio of the longest to shortest interval (scale). We show that when either of these quantities is poly-logarithmically bounded, we can approximate MIS for arbitrary length intervals.

Instead of assuming the max-degree or scale is bounded, we instead provide algorithms paramterized by these quantities. First, let the number of pair-wise intersections be bounded by $\kappa_{\max}$. Then,

▶ **Theorem 4** (Theorem 17, informal). *For $\epsilon > 0$, there exists a turnstile streaming algorithm that takes as input a set of unit-weight arbitrary-length intervals, with at most $\kappa_{\max}$ pair-wise intersections and with probability $99/100$, outputs a $(1+\epsilon)$-approximation to MIS in poly$(\log(n), \epsilon^{-1}, \kappa_{\max})$ space.*

This result requires several new algorithmic ideas. Observe, placing a unit grid no longer suffices since the intervals now span different lengths. Therefore, we impose a nested grid on our input, where the grid size is geometrically increasing, and randomly shift it. Further, observe that the natural strategy that partitions the interval into geometrically increasing *length classes* and estimates each partition up to $1+\epsilon$ does not work since the intervals overlap.

We therefore define the following object that uniquely determines intervals of a particular *length class* contributing to the MIS :

▶ **Definition 5** (r_i-Structure). *We define an r_i-Structure to be a subset of the Nested Grid, such that there exists an interval at the i^{th} grid level, there exist no intervals in the grid at any level $i' > i$ and all the intervals in the grid at levels $i' < i$ intersect the interval at the i^{th} level.*

It is easy to see any interval that contributes to MIS corresponds to an r_i-Structure for some i. Therefore, it suffices to estimate the number of r_i-Structures for all i. Following our approach for unit intervals, we again use k-Sparse Recovery as our main tool. At a high level, we sub-sample $\text{poly}(\log(n), \epsilon^{-1})$ r_i-Structures from the set of all such structures at level i, and create a new substream for each i. We then run a $\kappa_{\max}$-Sparse Recovery Algorithm on each substream. We show that at the end of the stream, we obtain an estimate of the number of r_i-Structures at level i that concentrates. Since the structures form a partition, our overall estimate is simply the sum of the estimates obtained for each i.

The main algorithmic challenge here is to show that we can indeed detect and subsample the r_i-structures. These structures are defined in a way that takes into account how many intervals appear in the nested grid both above and below a given interval. Therefore, it is unclear how to track such updates as they constantly change over the stream. However, observe that since our space is parameterized by the max-degree, we can afford to store an r_i-Structure completely in memory.

Given a randomly sampled cell from the i-th level of the nested grid, we assume this cell contributes an r_i-structure. We then run $\kappa_{\max}$-Sparse Recovery on this cell. Our main insight is that at the end of the stream we can verify whether this cell indeed contributed an r_i-structure since we recover the nested intervals *exactly*. The final remaining challenge is to ensure that our sub-sample contains a sufficient number of non-empty structures for each level and the resulting estimate concentrates. We describe these details in Section A.1.

Finally, we show that similar algorithmic ideas also result in a turnstile streaming algorithm, if parametrize the input by the $W_{\max}$, the ratio of the largest to smallest interval :

▶ **Theorem 6** (Theorem 20, informal). *For $\epsilon > 0$, there exists a turnstile streaming algorithm that takes as input a set of unit-weight arbitrary-length intervals, with $W_{\max}$ being an upper bound on the ratio of the largest to smallest interval, and with probability $99/100$, outputs a $(2+\epsilon)$-approximation to MIS in poly$(\log(n), \epsilon^{-1}, W_{\max})$ space.*

Hexagonal Packing of Circles in the Plane

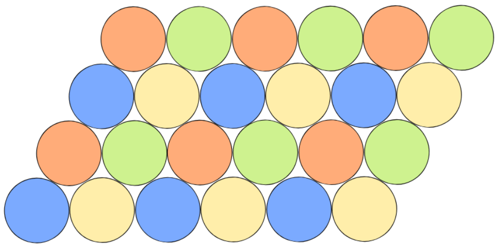

Figure 4.1 We illustrate the hexagonal circle packing in the Euclidean Plane. Each color represents an equivalence class. Observe that input disks that are centered in distinct circles of the same equivalence class are independent, since the circles are at least 2 units apart.

4.3 Unit-Radius Disks

We generalize the WMIS turnstile streaming algorithm for unit length intervals to unit radius disks in $\mathbb{R}^2$. The approximation ratio for disks is closely related to the optimal circle packing constant. We leverage the hexagonal packing of circles in the 2-D plane to obtain the following result:

▶ **Theorem 7** (Theorem 21, informal). *There exists a turnstile streaming algorithm achieving a $\left(\frac{8\sqrt{3}}{\pi}+\epsilon\right)$-approximation to estimate WMIS of unit disks with constant probability and in poly$\left(\frac{\log(n)}{\epsilon}\right)$ space.*

We note that a greedy algorithm for unweighted disks obtains a 5-approximation to α [25] and the space required is $O(\alpha)$. The greedy algorithm can be extended to obtain a $(5+\epsilon)$-approximation in $\text{poly}\left(\frac{\log n}{\epsilon}\right)$ space using the sampling approach we presented in Section 5. However, beating the approximation ratio achieved by the greedy algorithm requires geometric insight. Critically, we use the hexagonal packing of unit circles in a plane introduced by Lagrange , which was shown to be optimal by Toth [16].

The hexagonal packing covers a $\frac{\pi}{\sqrt{12}}$ fraction of the area in two dimensions. We then partition the unit circles in the hexagonal packing into equivalence classes such that two circles in the same equivalence class are at least a unit distance apart. Formally, let c_1, c_2 be two unit circles in the hexagonal packing of the plane lying in the same equivalence class. Then, for all points $p_1 \in c_1$, $p_i \in c_2$, $\|p_1 - p_2\|_2 \geq 1$. Therefore, if two input disks of unit radius have centers lying in distinct circles belong to the same equivalence class, the disks must be independent, as long as the disk are not centered on the boundary of the circles.

Randomly shifting the underlying hexagonal packing ensures this happens with probability 1. We then show that we can partition the hexagonal packing into four equivalence classes such that their union covers all the circles in the packing and disks lying in distinct circles of the same equivalence class are independent.

Algorithmically, we first impose a grid Δ of the hexagonal packing of circles with radius 1 and shift it by a random integer. We discard all disks that do not have centers lying inside the grid Δ. Given that a hexagonal packing covers a $\pi/\sqrt{12}$ fraction of the area, in expectation, we discard a $(1-\pi/\sqrt{12})$ fraction of $|\mathsf{OPT}|$. We note that if we could accurately estimate the remaining WMIS, and scale the estimator by $\sqrt{12}/\pi$, we would obtain a $(\sqrt{12}/\pi)$-approximation to $|\mathsf{OPT}|$. Let $||\mathsf{OPT}|_{\text{hp}}|$ denote the remaining WMIS. By Theorem 16 such an approximation requires $\Omega(n)$ space.

We then observe that the hexagonal circle packing grid can be partitioned into four equivalence classes. We use $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ and $\mathcal{C}_4$ to denote these equivalence classes. Since the equivalence classes form a partition of the hexagonal packing, at least one of them must contain a 1/4-fraction of the remaining maximum independent set. W.l.o.g, let $\mathcal{C}_1$ be the partition that contributes the most to $|\mathsf{OPT}|$. Then, $|\mathsf{OPT}_{\mathcal{C}_1}| \geq \frac{1}{4}|\mathsf{OPT}_{\mathrm{hp}}|$. Therefore, we focus on designing an estimator for $\mathcal{C}_1$. We show a $(1+\epsilon)$-approximation to $\mathcal{C}_1$ in $\mathrm{poly}\left(\log(n), \epsilon^{-1}\right)$ space generalizing the algorithmic ideas we introduced for Theorem 8. This implies an overall $\left(\frac{4\sqrt{12}}{\pi} + \epsilon\right) = \left(\frac{8\sqrt{3}}{\pi} + \epsilon\right)$ approximation for $|\mathsf{OPT}|$.

5 Weighted Interval Selection for Unit Intervals

In this section, we present an algorithm to approximate the weight of the maximum independent set, β, for unit-length intervals in turnstile streams. Interestingly, we note that estimating β has the same complexity as approximating α for unit-length intervals. That is, we obtain a $(2+\epsilon)$-approximation to β in the turnstile model, which immediately implies $(2+\epsilon)$-approximation for α, where the weights are identical. We complement this result with a lower bound that shows any $(2-\epsilon)$-approximation to α requires $\Omega(n)$ space. The main algorithmic guarantee we achieve is as follows:

▶ **Theorem 8.** *Let $\mathcal{P}$ be a turnstile stream of weighted unit intervals such that the weights are polynomially bounded in n and let $\epsilon \in (0, 1/2)$. There exists an algorithm that outputs an estimator Y such that with probability at least $9/10$ the following guarantees hold:*

1. $\frac{\beta}{2(1+\epsilon)} \leq Y \leq \beta$.
2. *The total space used is* $\mathrm{poly}\left(\frac{\log(n)}{\epsilon}\right)$.

We first impose a grid Δ of side length 1 and shift it by a random integer. We then snap each interval to the cell containing the center of the interval and partition the real line into odd and even cells. Let $\mathcal{C}_e$ be the set of all even cells and $\mathcal{C}_o$ be the set of all odd cells. By averaging, either $|\mathtt{OPT}_{\mathcal{C}_e}|$ or $|\mathtt{OPT}_{\mathcal{C}_o}|$ is at least $\frac{\beta}{2}$. We describe an estimator that gives a $(1+\epsilon)$-approximation to $|\mathtt{OPT}_{\mathcal{C}_e}|$ and $|\mathtt{OPT}_{\mathcal{C}_o}|$. W.l.o.g let $|\mathtt{OPT}_{\mathcal{C}_e}| \geq |\mathtt{OPT}_{\mathcal{C}_o}|$. Therefore, taking the max of the two estimators, we obtain a $(2+\epsilon)$-approximation to β.

Having reduced the problem to estimating $|\mathtt{OPT}_{\mathcal{C}_e}|$, we observe that each even cell has at most 1 interval, namely the max weight interval landing in the cell, contributing to $\mathtt{OPT}_{\mathcal{C}_e}$. Then, partitioning the cells in $\mathcal{C}_e$ into $\mathrm{poly}(\log(n))$ weight classes based on the max weight interval in each cell and approximately counting the number of cells in each weight class suffices to estimate $|\mathtt{OPT}_{\mathcal{C}_e}|$ up to a $(1+\epsilon)$-factor. Given such a partition, we can create a substream for each weight class in the partition and compute the ℓ_0 norm of each substream. However, we do not know the partition of the cells into the weight classes a priori and this partition can vary drastically over the course of stream given that intervals can be deleted. The main technical challenge is to simulate this partition. A key tool we use is to estimate the ℓ_0 norm of a vector in turnstile streams. Kane, Nelson and Woodruff [40] showed how to obtain a $(1 \pm \epsilon)$-approximation to the ℓ_0-norm of a vector in $\mathrm{poly}(\frac{\log(n)}{\epsilon})$ space.

▶ **Theorem 9** (ℓ_0-Norm Estimation [40])**.** *In the turnstile model, there is an algorithm for $(1 \pm \epsilon)$-approximating the ℓ_0-norm (number of non-zero coordinates) of a vector using space* $\mathrm{poly}\left(\frac{\log(n)}{\epsilon}\right)$ *with success probability $2/3$.*

We begin by describing a simple algorithm which obtains a weaker $(9/2+\epsilon)$-approximation to $|\mathtt{OPT}_{\mathcal{C}_e}|$ and in turn a $(9+\epsilon)$-approximation to β. Formally, consider a partition of cells in $\mathcal{C}_e$ into $b = \mathrm{poly}(\log(n))$ weight classes $\mathcal{W}_i = \{c \in \mathcal{C}_e | (1+1/2)^i \leq m(c) < (1+1/2)^{i+1}\}$, where $m(c)$ is the maximum weight of an interval in c. Create a substream for each weight class $\mathcal{W}_i$, denoted by $\mathcal{W}'_i$, and feed an input interval into this substream if its weight lies in the range $[(1+1/2)^i, (1+1/2)^{i+1})$. Let t_i be the corresponding ℓ_0 estimate for substream $\mathcal{W}'_i$. Then, we can approximate the contribution of $\mathcal{W}_i$ by $(1+1/2)^{i+1} \cdot t_i$. Summing over the b weight classes gives an estimate for $|\texttt{OPT}_{\mathcal{C}_e}|$.

Given access to an algorithm for estimating the ℓ_0-norm, the Naïve Approximation Algorithm (1) satisfies the following guarantee:

▶ **Lemma 10.** *The Naïve Approximation Algorithm (1) outputs an estimate X such that with probability $99/100$, $\frac{\beta}{9(1+\epsilon)} \leq X \leq \beta$ and runs in space $\text{poly}\left(\frac{\log(n}{\epsilon}\right)$.*

Proof. We observe that for each non-empty cell $c \in \mathcal{C}_e$, there is exactly 1 interval that can contribute to $|\texttt{OPT}_{\mathcal{C}_e}|$ since each cell of the grid has side length 1 and all intervals falling in a given cell pairwise intersect. This contributing interval lies in some weight class $\mathcal{W}_i$ and our estimator approximates its weight as $(1+1/2)^{i+1}$. Here, the weights of the intervals are sandwiched between $(1+1/2)^i$ and $(1+1/2)^{i+1}$. Therefore, we overestimate the weight by a factor of at most $3/2$.

Further, instead of taking the maximum over each cell c, we may have inserted intervals that lie in c into all substreams $\mathcal{W}'_i$. Therefore, we take the sum of our geometrically increasing weight classes over that cell. In the worst case, we approximate the true weight of a contributing interval, $(3/2)^{i+1}$, with $\sum_{i'=1}^{i} (3/2)^{i'+1} = 3((3/2)^{i+1} - 1)$. Note, we again overestimate the weight, this time by a factor of 3.

Finally, Theorem 9 overestimates the ℓ_0-norm of $\mathcal{W}_i$ by at most $1+\epsilon$ with probability at least $2/3$. We boost this probability by running $O(\log(n))$ estimators and taking the median. Union bounding over all $i \in [b]$, we simultaneously overestimate the ℓ_0-norm of all $\mathcal{W}_i$ by at most $1+\epsilon$ with probability at least $99/100$. Therefore, the overall estimator is a $(9/2+\epsilon)$-approximation to $|\texttt{OPT}_{\mathcal{C}_e}|$. Rescaling our estimator by the above constant underestimates $|\texttt{OPT}_{\mathcal{C}_e}|$. Finally, $|\texttt{OPT}_{\mathcal{C}_e}| \geq \beta/2$ and $\frac{\beta}{(9+\epsilon)} \leq X \leq \beta$.

Since our weights are polynomially bounded, we create $\text{poly}\big(\log_{1+\epsilon}(n)\big)$ substreams and run an ℓ_0 estimator from Theorem 9 on each substream. Therefore, the total space used by Algorithm 1 is $\text{poly}\left(\frac{\log(n}{\epsilon}\right)$. ◀

We can thus assume we know β and $|\texttt{OPT}_{\mathcal{C}_e}|$ up to a constant by initially making $O(\log(n))$ guesses and running the Naïve Approximation Algorithm for each guess in parallel. At the end of the stream, we know the correct guess up to a constant factor, and thus can output the estimator corresponding to that branch of computation. A key tool we use in this algorithm is k-`Sparse Recovery`. As mentioned in Berinde et al. [8], the k-tail guarantee is a sufficient condition for k-`Sparse Recovery`, since in a k-sparse vector, the elements of the tail are 0. We note that the `Count-Sketch` Algorithm [17] has a k-tail guarantee in turnstile streams.

▶ **Definition 11** (k-`Sparse Recovery`)**.** *Let x be the input vector such that x is updated coordinate-wise in the turnstile streaming model. Then, x is k-sparse if x has at most k non-zero entries at the end of the stream of updates. Given that x is k-sparse, a data structure that exactly recovers x at the end of the stream is referred to as a k-`Sparse Recovery` data structure.*

Intuitively, we again simulate partitioning cells in $\mathcal{C}_e$ into $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ weight classes according to the maximum weight occurring in each cell. Since we do not know this partition a priori, we initially create $b = O\left(\frac{\log(n)}{\epsilon}\right)$ substreams, one for each weight class and run the

ℓ_0-estimator on each one. We then make $O\left(\frac{\log(n)}{\epsilon}\right)$ guesses for $|\mathtt{OPT}_{\mathcal{C}_e}|$ and run the rest of the algorithm for each branch in parallel. Additionally, we run the Naïve Approximation Algorithm to compute the right value of $|\mathtt{OPT}_{\mathcal{C}_e}|$ up to a constant factor, which runs in space poly $\left(\frac{\log(n)}{\epsilon}\right)$. Then, we create $b = \text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ substreams by agnostically sampling cells with probability $p_i = \Theta\left(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X}\right)$, where X is the right guess for $|\mathtt{OPT}_{\mathcal{C}_e}|$. Sampling at this rate preserves a sufficient number of cells from weight class $\mathcal{W}_i$. We then run a sparse recovery algorithm on the resulting substreams.

We note that the resulting substreams are sparse. To see this, note we can filter out cells that belong weight classes $\mathcal{W}_{i'}$ for $i' < i$ by simply checking if the maximum interval seen so far lies in weight classes $\mathcal{W}_i$ and higher. Further, sampling with probability proportional to $\Theta\left(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 |\mathtt{OPT}_{\mathcal{C}_e}|}\right)$ ensures that the number of cells that survive from weight classes $\mathcal{W}_i$ and above are small. Therefore, we recover all such cells using the sparse recovery algorithm. Note, we limit the algorithm to considering weight classes that have a non-trivial contribution to $\mathtt{OPT}_{\mathcal{C}_e}$.

Using the ℓ_0 norm estimates computed above, we can determine the number of non-empty cells in each of the weight classes. Thus, we create a threshold for weight classes that contribute, such that all the weight classes below the threshold together contribute at most an ϵ-fraction of $|\mathtt{OPT}_{\mathcal{C}_e}|$ and we can set their corresponding estimators to 0. Further, for all the weight classes above the threshold, we can show that sampling at the right rate leads to recovering enough cells to achieve concentration in estimating their contribution.

Next, we show that the total space used by Algorithm 2 is poly $\left(\frac{\log(n)}{\epsilon}\right)$. We initially create $b = O\left(\frac{\log(n)}{\epsilon}\right)$ substreams, one for each weight class and run an ℓ_0-estimator on each one. Recall, this requires poly $\left(\frac{\log(n)}{\epsilon}\right)$. We then make $O\left(\frac{\log(n)}{\epsilon}\right)$ guesses for $|\mathtt{OPT}_{\mathcal{C}_e}|$ and run the rest of the algorithm for each branch in parallel. Additionally, we run Algorithm 1 to compute the right value of $|\mathtt{OPT}_{\mathcal{C}_e}|$ up to a constant factor, which runs in space poly $\left(\frac{\log(n)}{\epsilon}\right)$. Then, we create b substreams by sampling cells with probability $p_i = \Theta\left(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X}\right)$, for $i \in [b]$. Subsequently, we run a poly $\left(\frac{\log(n)}{\epsilon}\right)$-sparse recovery algorithm on each one. Note, if each sample is not too large, this can be done in poly $\left(\frac{\log(n)}{\epsilon}\right)$ space. Therefore, it remains to show that each sample $\mathcal{S}_i$ is small.

▶ **Lemma 12.** *Given a turnstile stream $\mathcal{P}$, with probability at least $99/100$, the Weighted Unit Interval Turnstile Sampling procedure (Algorithm 2) samples poly $\left(\frac{\log(n)}{\epsilon}\right)$ cells from the grid Δ.*

Proof. For $i \in [b]$, let $\mathcal{S}_i$ be a substream of cells in $\mathcal{C}_e$, sampled with probability p_i and having an interval with weight at least $(1+\epsilon)^i$ since we filter out all cells with smaller weight. Then, by an averaging argument, the total number of cells with an interval of weight at least $(1+\epsilon)^i$ is at most $\frac{\beta}{(1+\epsilon)^i}$. Sampling with probability $p_i = \Theta\left(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X}\right)$, the expected number of cells from $\mathcal{W}_i$ that survive in $\mathcal{S}_i$ is at most $p_i \frac{\beta}{(1+\epsilon)^i} = \text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ in expectation. Next, we show that they are never much larger than their expectation. Let X_c be the indicator random variable for cell $c \in \mathcal{W}_i$ to be sampled in $\mathcal{S}_i$ and let μ be the expected number of cells in $\mathcal{S}_i$. By Chernoff bounds,

$$\Pr\left[\sum_c X_c \geq (1+\epsilon)\mu\right] \leq \exp\left(-\frac{2\epsilon^2 \text{poly}(\log(n))}{\text{poly}(\epsilon)}\right) \leq 1/n^k$$

for some large constant k. A similar argument holds for the number of cells from weight class $\mathcal{W}_{i'}$, for $i' > i$, surviving in substream $\mathcal{S}_i$. Note, for all $i' < i$, we never include such a cell from weight class $\mathcal{W}_{i'}$ in our sample $\mathcal{S}_i$, since the filtering step rejects all cells that do not contain an interval of weight at least $(1+\epsilon)^i$. Union bounding over the events that cells $c \in \mathcal{W}_{i'}$ get sampled in $\mathcal{S}_i$, for $i' \geq i$, the cardinality of $\mathcal{S}_i$ is at most $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ with probability at least $1 - 1/n^{k'}$ for an appropriate constant k'. Since we create b such substreams for $\mathcal{C}_e$, we can union bound over such events in each of them and thus $\bigcup_{i\in[b]} |\mathcal{S}_i|$ is at most $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ with probability at least $99/100$. Since $|\mathcal{C}_e|$ is $|\Delta|/2$, the same result holds for the total cells sampled from Δ. Therefore, the overall space used by Algorithm 2 is $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$. ◀

Algorithm 2 Weighted Unit Interval Turnstile Sampling.

Input: Given a turnstile stream $\mathcal{P}$ with weighted unit intervals, where the weights are polynomially bounded, ϵ and $\delta > 0$, the sampling procedure outputs a $(2+\epsilon)$-approximation to β.

1. Randomly shift a grid Δ of side length 1. Partition the cells into $\mathcal{C}_e$ and $\mathcal{C}_o$.
2. For cells in $\mathcal{C}_e$, snap each interval in the input to a cell c that contains its center. Consider a partitioning of the cells in $\mathcal{C}_e$ into $b = \text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ weight classes $\mathcal{W}_i = \{c \in \mathcal{C}_e | (1+\epsilon)^i \leq m(c) \leq (1+\epsilon)^{i+1}\}$, where $m(c)$ is the maximum weight of an interval in c (we do not know this partition a priori.) Create a substream for each weight class $\mathcal{W}_i$ denoted by $\mathcal{W}'_i$.
3. Feed interval $D(d_j, 1, w_j)$ along substream $\mathcal{W}'_i$ such that $w_j \in [(1+\epsilon)^i, (1+\epsilon)^{i+1})$. Maintain a $(1 \pm \epsilon)$-approximate ℓ_0-estimator for each substream. Let $|\mathcal{W}'_i|$ denote the number of non-empty cells in substream $\mathcal{W}'_i$ and $X_{\mathcal{W}'_i}$ be the corresponding estimate returned by the ℓ_0-estimator.
4. Create $O(\log(n))$ substreams, one for each guess of $|\texttt{OPT}_{\mathcal{C}_e}|$. Let X be the guess for the current branch of the computation. In parallel, run Algorithm 1 estimates $|\texttt{OPT}_{\mathcal{C}_e}|$ up to a constant factor. Therefore, at the end of the stream, we know a constant factor approximation to the correct value of $|\texttt{OPT}_{\mathcal{C}_e}|$ and use the estimator from the corresponding branch of the computation.
5. In parallel, for $i \in [b]$, create substream $\mathcal{S}_i$ by subsampling cells in $\mathcal{C}_e$ with probability $p_i = \Theta\left(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X}\right)$. Note, this sampling is done agnostically at the start of the stream.
6. Run a $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$-sparse recovery algorithm on each substream $\mathcal{S}_i$. For substream $\mathcal{S}_i$, filter out cells c such that $m(c) < (1+\epsilon)^i$. Let $\mathcal{S}'_i$ be the set of cells recovered by the sparse recovery algorithm. Let $\mathcal{S}'_{i|\mathcal{W}_i}$ be the cells in $\mathcal{S}'_i$ that belong to weight class $\mathcal{W}_i$.
7. Let $X_{\mathcal{W}'} = \sum_{i\in[b]} X_{\mathcal{W}'_i}$. Let Z_c be a random variable such that $Z_c = \frac{(1+\epsilon)^{i+1}}{p_i}$ if $c \in \mathcal{S}'_{i|\mathcal{W}_i}$ and 0 otherwise. If $X_{\mathcal{W}'_i} \geq \frac{\epsilon X_{\mathcal{W}'}}{(1+\epsilon)^{i+1} b}$, set the estimator for the i^{th} subsample, $Y_i = \sum_{c\in\mathcal{S}'_{i|\mathcal{W}_i}} X_{\mathcal{W}'_i} Z_c / \mathcal{S}'_{i|\mathcal{W}_i}$. Otherwise, set $Y_i = 0$. Let $Y_e = \sum_i Y_i$.
8. Repeat Steps 2-7 for the set $\mathcal{C}_o$ and let Y_o be the corresponding estimator.

Output: $Y = \max(Y_e, Y_o)$.

Next, we show that the estimate returned by our sampling procedure is indeed a $(2+\epsilon)$-approximation. We observe that the union of the $\mathcal{W}_i$'s form a partition of $\mathcal{C}_e$. Therefore, it suffices to show that we obtain a $(1+\epsilon)$-approximation to the WIS for each $\mathcal{W}_i$ with good probability. Let c denote a cell in $\mathcal{W}_i$ and $\texttt{OPT}_c$ denote the $\texttt{WIS}$ in cell c. We create a substream for each weight class $\mathcal{W}_i$ denoted by $\mathcal{W}_i'$ and let $X_{\mathcal{W}_i'}$ be the corresponding estimate returned by the ℓ_0 norm of $\mathcal{W}_i'$. Let $X_{\mathcal{W}'} = \sum_{i \in [b]} X_{\mathcal{W}_i'}$ denote the sum of the estimates across the b substreams.

We say that weight class $\mathcal{W}_i$ contributes if $X_{\mathcal{W}_i'} \geq \frac{\epsilon X_{\mathcal{W}'}}{(1+\epsilon)^{i+1} b}$. Note, if we discard all the weight classes that do not contribute we lose at most an ϵ-fraction of β (as shown below). Therefore, setting the estimators corresponding to classes that do not contribute to 0 suffices. The main technical hurdle remaining is to show that if a weight class contributes we can accurately estimate $|\texttt{OPT}_{\mathcal{W}_i}|$.

▶ **Lemma 13.** *Let $Y_e = \sum_i Y_i$ be the estimator returned by Algorithm 2 for the set $\mathcal{C}_e$. Then, $Y_e = (1 \pm \epsilon)|\texttt{OPT}_{\mathcal{C}_e}|$ with probability at least $99/100$.*

Proof. We first consider the case when $\mathcal{W}_i$ contributes, i.e., $X_{\mathcal{W}_i'} \geq \frac{\epsilon X_{\mathcal{W}'}}{(1+\epsilon)^{i+1} b}$. Note, $X_{\mathcal{W}'} = \sum_{i \in [b]} X_{\mathcal{W}_i'}$ is a $(1 \pm \epsilon)$-approximation to the number of non-empty cells in $\mathcal{W}$ with probability at least $1 - n^{-k}$, where $\mathcal{W} = \bigcup_{i \in [b]} \mathcal{W}_i$, since the ℓ_0-estimator is a $(1 \pm \epsilon)$-approximation to the number of non-empty cells in $\mathcal{W}_i$ simultaneously for all i with high probability and the $\mathcal{W}_i$'s are disjoint. Recall, X is the correct guess for $|\texttt{OPT}_{\mathcal{C}_e}|$. Therefore, $(1+\epsilon)^i X_{\mathcal{W}_i'} = \Omega\left(\frac{\epsilon X}{(1+\epsilon)b}\right)$. Then, sampling at a rate $p_i = \Theta(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X})$ implies at least $\Omega\left(\frac{\epsilon X}{(1+\epsilon)^{i+1} b}\right) \cdot \Theta(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X}) = \Omega\left(\frac{\log(n)}{(1+\epsilon)\epsilon^2}\right)$ cells from $\mathcal{W}_i$ survive in expectation. Let X_c denote an indicator random variable for cell $c \in \mathcal{W}_i$ being in substream $\mathcal{S}_i$. Then, by a Chernoff bound,

$$\Pr\left[\sum_{c \in \mathcal{W}_i} X_c \leq (1-\epsilon)\left(\frac{\log(n^{-c})}{2\epsilon^2}\right)\right] \leq \exp\left(\frac{-2\epsilon^2 \log(n^{-c})}{2\epsilon^2}\right) \leq n^{-c}$$

for some constant c. Union bounding over all the random events similar to the one above for $i \in [b]$, simultaneously for all i, the number of cells from $\mathcal{W}_i$ in $\mathcal{S}_i$ is at least $\Omega\left(\frac{\log(n)}{\epsilon^2}\right)$ with probability at least $1 - 1/n^k$ for some constant k. Note, for $i' < i$, no cell $c \in \mathcal{W}_{i'}$ exists in $\mathcal{S}_i$ since the filter step removes all cells c such that $m(c) < (1+\epsilon)^i$.

Next, consider a weight class $\mathcal{W}_{i'}$ for $i' > i$ such that it contributes. We upper bound the number of cells from $\mathcal{W}_{i'}$ that survive in substream $\mathcal{S}_i$. Note, weight class $\mathcal{W}_{i'}$ contains at most $\frac{\beta}{(1+\epsilon)^{i+1}}$ non empty cells for $i' > i$. In expectation, at most $\frac{\beta}{(1+\epsilon)^{i+1}} \cdot p_i = O\left(b \frac{\log(n)}{(1+\epsilon)\epsilon^3}\right)$ cells from $\mathcal{W}_{i'}$ survive in sample $\mathcal{S}_i$, for $i' > i$. By a Chernoff bound, similar to the one above, simultaneously for all $i' > i$, at most $O\left(b \frac{\log(n)}{(1+\epsilon)\epsilon^3}\right)$ cells from $\mathcal{W}_{i'}$ survive, with probability at least $1 - 1/n^{k'}$.

Now, we observe that the total number of cells that survive the sampling process in substream $\mathcal{S}_i$ is poly$\left(\frac{\log(n)}{\epsilon}\right)$ and therefore, they can be recovered exactly by the poly$\left(\frac{\log(n)}{\epsilon}\right)$-sparse recovery algorithm. Let the resulting set be denoted by $\mathcal{S}_i'$. We can also compute the number of cells that belong to weight class $\mathcal{W}_i$ that are recovered in the set $\mathcal{S}_i'$ and we denote this by $|\mathcal{S}_{i|\mathcal{W}_i}'|$. Recall, the corresponding estimator is $Y_i = \sum_{c \in \mathcal{S}_{i|\mathcal{W}_i}'} \frac{X_{\mathcal{W}_i'} Z_c}{|\mathcal{S}_{i|\mathcal{W}_i}'|}$, where $Z_c = \frac{(1+\epsilon)^{i+1}}{p_i}$ if $c \in \mathcal{S}_{i|\mathcal{W}_i}'$ and 0 otherwise. We first show we obtain a good estimator for $|\texttt{OPT}_{\mathcal{W}_i}|$ in expectation: $\mathbf{E}[Y_i] = \mathbf{E} \sum_{c \in \mathcal{S}_{i|\mathcal{W}_i}'} X_{\mathcal{W}_i'} Z_c / \mathcal{S}_{i|\mathcal{W}_i}' = (1 \pm 4\epsilon)|\texttt{OPT}_{\mathcal{W}_i}|$.

Since we know that $|\mathcal{S}_{i|\mathcal{W}_i}'| = \Omega(\frac{\log(n)}{(1+\epsilon)\epsilon^2})$, we show that our estimator concentrates. Note, $\mathbf{E}[Y_i] = (1+\epsilon)^{i+1} X_{\mathcal{W}_i'} = \Omega(\frac{\epsilon X}{\log(n)})$. Further, $0 \leq Z_c \leq \frac{(1+\epsilon)^{i+1}}{p_i} = O\left(\frac{(1+\epsilon)^{i+1}\epsilon^3 X}{b(1+\epsilon)^i \log(n)}\right)$. By a Hoeffding bound, $\Pr\Big[|Y_i - E[Y_i]| \geq \epsilon E[Y_i]\Big] \leq 2\exp\left(\frac{\Omega(\log(n))}{1+\epsilon}\right) \leq 1/n^k$ for some constant

k. Therefore, union bounding over all i, Y_i is a $(1 \pm \epsilon)^2$-approximation to $|\mathtt{OPT}_{\mathcal{W}_i}|$ with probability at least $1 - 1/n$. Therefore, if $\mathcal{W}_i$ contributes we obtain a $(1 \pm \epsilon)$-approximation to $|\mathtt{OPT}_{\mathcal{W}_i}|$.

In the case where $\mathcal{W}_i$ does not contribute, we set the corresponding estimator to 0. Note, $X_{\mathcal{W}_i'} < \frac{\epsilon X_{\mathcal{W}'}}{(1+\epsilon)^{i+1} b} = \frac{\epsilon(1\pm\epsilon)|\mathtt{OPT}_{\mathcal{W}}|}{b} = O(\frac{\epsilon\beta}{b})$. Note, since there are at most b weight classes, discarding all weight classes that do not contribute discards at most $O(\epsilon\beta)$. We therefore lose at most an ϵ-fraction of β by setting the Y_i corresponding to non-contributing weight classes to 0. ◀

Combining Lemmas 10 and 13 finishes the proof for Theorem 8.

5.1 Lower bound for Unit Intervals

Here, we describe a communication complexity lower bound for estimating α for unit-length interval in turnstile streams and thus show the optimality of Theorem 8. Our starting point is the `Augmented Index` problem and its communication complexity is well understood in the two-player one-way communication model. In this model, we have two players Alice and Bob who are required to compute a function based on their joint input and Alice is allowed to send messages to Bob that are a function of her input and finally Bob announces the answer. Note, Bob isn't allowed to send messages to Alice.

▶ **Definition 14** (`Augmented Indexing`). *Let $\mathtt{AI}_{n,j}$ denote the communication problem where Alice receives as input $x \in \{0,1\}^n$ and Bob receives an index $j \in [n]$, along with the $x_{j'}$ for $j' > j$. The objective is for Bob to output x_j in the one-way communication model.*

▶ **Theorem 15** (Communication Complexity of $\mathtt{AI}_{n,j}$, [47]). *The randomized one-way communication complexity of $\mathtt{AI}_{n,j}$ with error probability at most $1/3$ is $\Omega(n)$.*

Let `Alg` be a one-pass turnstile streaming algorithm that estimates α. We show that `Alg` can be used as a subroutine to solve $\mathtt{AI}_{n,j}$, in turn implying a lower bound on the space complexity of `Alg`. We formalize this idea in the following theorem:

▶ **Theorem 16.** *Any randomized one-pass turnstile streaming algorithm `Alg` which approximates α to within a $(2-\epsilon)$-factor, for any $\epsilon > 0$, for unit intervals, with at least constant probability, requires $\Omega(n)$ space.*

Proof. Given her input x, Alice constructs a stream of unit-length intervals and runs `Alg` on the stream. For $i \in [n]$, Alice inserts the interval $\left[\frac{2i-x_i}{n^2}, (\frac{2i-x_i}{n^2}) + 1\right]$. She then communicates the state of `Alg` to Bob. Bob uses the message received from Alice as the initial state of the algorithm and continues the stream. Since Bob's input includes an index j and x_i for all $i > j$, Bob deletes all intervals corresponding to such i. Bob then inserts $\left[(\frac{2j-0.5}{n^2}) - 1, \frac{2j-0.5}{n^2}\right]$.

Let us consider the case where $x_j = 1$. We first note that Bob's interval is the leftmost interval in the remaining set. The right endpoint of this interval is $\frac{2j-0.5}{n^2}$. Next, the rightmost interval corresponds to the j^{th} interval inserted by Alice. The left endpoint of this interval is $\frac{2j-1}{n^2}$. Clearly, these intervals intersect each other and intersect all the intervals between them. Therefore, $\alpha = 1$.

Let us now consider the case where $x_j = 0$. Again, Bob's interval is the leftmost with its right endpoint at $\frac{2j-0.5}{n^2}$. However, the left endpoint of Alice's rightmost interval is $\frac{2j}{n^2}$ and thus these two intervals are independent. Therefore, $\alpha \geq 2$. Observe, any $(2-\epsilon)$-approximate algorithm can distinguish between these two cases and solve $\mathtt{AI}_{n,j}$. By Theorem 15, any such algorithm requires $\Omega(n)$ communication and in turn $\Omega(n)$ space. ◀

References

1 Pankaj K Agarwal, Marc Van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry*, 11(3-4):209–218, 1998.

2 Yuqing Ai, Wei Hu, Yi Li, and David P. Woodruff. New characterizations in turnstile streams with applications. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 20:1–20:22, 2016. doi:10.4230/LIPIcs.CCC.2016.20.

3 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29. ACM, 1996.

4 Baruch Awerbuch, Yair Bartal, Amos Fiat, and Adi Rosén. Competitive non-preemptive call control. In *SODA*, volume 94, pages 312–320, 1994.

5 Yossi Azar and Oren Gilon. Buffer management for packets with processing times. In *Algorithms-ESA 2015*, pages 47–58. Springer, 2015.

6 Unnar Th Bachmann, Magnús M Halldórsson, and Hadas Shachnai. Online scheduling intervals and t-intervals. *Full version*, 2010.

7 Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 1–10. Springer, 2002.

8 Radu Berinde, Graham Cormode, Piotr Indyk, and Martin J. Strauss. Space-optimal heavy hitters with strong error bounds. In *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, pages 157–166, 2009. doi:10.1145/1559795.1559819.

9 Jarosław Błasiok. Optimal streaming and tracking distinct elements with high probability. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2432–2448. SIAM, 2018.

10 Sergio Cabello and Pablo Pérez-Lantero. Interval selection in the streaming model. In *Algorithms and Data Structures - 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5-7, 2015. Proceedings*, pages 127–139, 2015. doi:10.1007/978-3-319-21840-3_11.

11 Sergio Cabello and Pablo Pérez-Lantero. Interval selection in the streaming model. *Theoretical Computer Science*, 702:77–96, 2017.

12 Ran Canetti and Sandy Irani. Bounding the power of preemption in randomized scheduling. *SIAM Journal on Computing*, 27(4):993–1015, 1998.

13 Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 892–901. Society for Industrial and Applied Mathematics, 2009.

14 Timothy M Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *Journal of Algorithms*, 46(2):178–189, 2003.

15 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.

16 Hai-Chau Chang and Lih-Chung Wang. A simple proof of thue's theorem on circle packing. *arXiv preprint*, 2010. arXiv:1009.4322.

17 Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, pages 693–703, 2002. doi:10.1007/3-540-45465-9_59.

18 Julia Chuzhoy and Alina Ene. On approximating maximum independent set of rectangles. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 820–829. IEEE, 2016.

19 Brent N Clark, Charles J Colbourn, and David S Johnson. Unit disk graphs. *Discrete mathematics*, 86(1-3):165–177, 1990.

20 Peter Clifford and Ioana Cosma. A simple sketching algorithm for entropy estimation over streaming data. In *Artificial Intelligence and Statistics*, pages 196–206, 2013.

21 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

22 Graham Cormode, Jacques Dark, and Christian Konrad. Independent set size approximation in graph streams. *CoRR*, abs/1702.08299, 2017. `arXiv:1702.08299`.

23 Graham Cormode, Jacques Dark, and Christian Konrad. Independent sets in vertex-arrival streams. *arXiv preprint*, 2018. `arXiv:1807.08331`.

24 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. `doi:10.1007/978-3-642-31594-7_26`.

25 Thomas Erlebach and Jirı Fiala. The maximum independent set problem in unit disk graphs, 2003.

26 Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing*, 34(6):1302–1323, 2005.

27 Cristian Estan, George Varghese, and Mike Fisk. Bitmap algorithms for counting active flows on high speed links. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 153–166. ACM, 2003.

28 Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156. Discrete Mathematics and Theoretical Computer Science, 2007.

29 Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

30 Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal packing and covering in the plane are np-complete. *Information processing letters*, 12(3):133–137, 1981.

31 Fănică Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, 1972.

32 Phillip B Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pages 281–291. ACM, 2001.

33 William K Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

34 Johan Håstad. Clique is hard to approximate within $\mathrm{n}^{\text{1-epsilon}}$. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 627–636, 1996. `doi:10.1109/SFCS.1996.548522`.

35 Petr Hliněnỳ. Contact graphs of line segments are np-complete. *Discrete Mathematics*, 235(1-3):95–106, 2001.

36 Dorit S Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

37 Hiroshi Imai and Takao Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of algorithms*, 4(4):310–323, 1983.

38 Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 49–58. ACM, 2011.

39 Daniel M Kane, Jelani Nelson, and David P Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1161–1178. Society for Industrial and Applied Mathematics, 2010.

40 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 41–52, 2010. doi:10.1145/1807085.1807094.
41 Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson Education India, 2006.
42 Antoon WJ Kolen, Jan Karel Lenstra, Christos H Papadimitriou, and Frits CR Spieksma. Interval scheduling: A survey. *Naval Research Logistics (NRL)*, 54(5):530–543, 2007.
43 Christian Konrad. Maximum matching in turnstile streams. In *Algorithms-ESA 2015*, pages 840–852. Springer, 2015.
44 Ping Li and Cun-Hui Zhang. A new algorithm for compressed counting with applications in shannon entropy estimation in dynamic data. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 477–496, 2011.
45 Ewa Malesinska. *Graph theoretical models for frequency assignment problems*. Shaker, 1997.
46 Andrew McGregor. Graph stream algorithms: a survey. *ACM SIGMOD Record*, 43(1):9–20, 2014.
47 Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.*, 57(1):37–49, 1998. doi:10.1006/jcss.1998.1577.
48 Kevin Mote. Fast point-feature label placement for dynamic visualizations. *Information Visualization*, 6(4):249–260, 2007.
49 Xiaoming Sun, David P Woodruff, Guang Yang, and Jialin Zhang. Querying a matrix through matrix-vector products. *arXiv preprint*, 2019. arXiv:1906.05736.
50 Gerhard J Woeginger. On-line scheduling of jobs with fixed start and end times. *Theoretical Computer Science*, 130(1):5–16, 1994.

A Arbitrary Length Intervals in Turnstile Streams

We now focus on estimating α and β for arbitrary-length intervals in turnstile streams. While we cannot obtain streaming algorithms in general, we show it is possible to estimate α and β when the maximum degree of the interval intersection graph or the maximum length of an interval arre bounded. In particular, we show an algorithm that achieves a $(1+\epsilon)$-approximation to α given that the maximum degree is upper bounded by $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$. We also parameterize the problem with respect to the maximum length of an interval, W_{max} (assuming the minimum length is 1), and give an algorithm using $\text{poly}\left(W_{max}\frac{\log(n)}{\epsilon}\right)$ space.

A.1 Algorithms under Bounded Degree Assumptions

In light of the lower bound, we study the problem of estimating α for arbitrary-length intervals assuming the number of pair-wise intersections are bounded by $\kappa_{\max} = \text{poly}\left(\frac{\log(n)}{\epsilon}\right)$. In this section we show the following theorem:

▶ **Theorem 17.** *Let $\mathcal{P}$ be an turnstile stream of unit-weight arbitrary-length intervals with lengths polynomially bounded in n and let $\epsilon \in (0, 1/2)$. Let $\kappa_{\max} = poly\left(\frac{\log(n)}{\epsilon}\right)$ be the maximum number of pairwise intersections in $\mathcal{P}$. Then, there exists an algorithm that outputs an estimator Y such that the following guarantees hold:*

1. $\frac{\alpha}{(1+\epsilon)} \leq Y \leq \alpha$ *with probability at least* $2/3$.
2. *The total space used is* $poly\left(\frac{\log(n)}{\epsilon}\right)$.

Algorithm 3 Level Estimator.

Input: Given a turnstile stream $\mathcal{P}$ with unit weight arbitrary length intervals, where the length is polynomially bounded, $\epsilon > 0$ and $\delta > 0$, the algorithm outputs a $(1+\epsilon)$-approximation to α, assuming that $\kappa_{\max} = \text{poly}\left(\frac{log(n)}{\epsilon}\right)$.

1. Let $t = O\left(\frac{\log(n)}{\epsilon}\right)$ be the number of level-classes. Let $\Delta = \bigcup_{i\in[t]} \Delta_i$ be a randomly shifted Nested Grid, where Δ_i is a grid of side length $\frac{(1+\epsilon)^{i+1}}{\epsilon}$.
2. For $i \in [t]$, let $\mathcal{R}_i$ be the set of all r_i-Structures at level i, where a r_i-Structure is a subset of the Nested Grid, Δ, such that there exists an interval at the i^{th} level of the structure, there exist no intervals in the structure at any level $i' > i$ and all the intervals in the structure at levels $i' < i$ intersect the interval at the i^{th} level.
3. For all $i \in [t]$, using Algorithm 4, sample $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ r_i-Structures from the set $\mathcal{R}_i$ to create a substream R_i^s. Note, this sampling is carried out with probability p_i defined below.
4. At the end of the stream, we recover R_i^s, for all $i \in [t]$. Let $Y_i = \frac{|\texttt{OPT}_{R_i^s}|}{p_i}$ (where p_i is the sampling probability for the i^{th} level), where $|\texttt{OPT}_{R_i^s}|$ can be computed using an offline algorithm.

Output: $Y = \sum_{i\in[t]} Y_i$.

Let W be the maximum length of the intervals in our input. We split our input into $t = O\left(\frac{\log(n)}{\epsilon}\right)$ length classes $\mathcal{W}_i$ such that for all $i \in [t]$, $\mathcal{W}_i = \{D_j \in \mathcal{P} | (1+\epsilon)^i \leq r_j \leq (1+\epsilon)^{i+1}\}$. Let $\mathcal{W}$ denote $\bigcup_{i\in[t]} \mathcal{W}_i$. We note that the partition here is over the input to the problem.

We can estimate the number of non-empty cells in each weight class up to a $(1\pm\epsilon)$-factor by creating a substream for each one and running an ℓ_0 estimator on them. At the end of the stream, we can discard classes that are not within $\log(W)$ non-empty cells of each other. Therefore, we can assume the remaining classes have the same number of non-empty cells up to a $\log(W)$ factor.

We then make $O(\log(n))$ guesses for the number of non-empty cells for any fixed level and run our algorithm in parallel for each guess. Since there are t levels, this gives rise to an $O(t\log(n))$ factor blowup in space. At the end of the stream we know the correct value for each level via the ℓ_0 estimates. Let the number of non-empty cells at every level be denoted by X_i.

In contrast with our previous algorithm, we note that placing a grid on the input with side length 1 no longer suffices since our intervals may now lie in multiple cells. Therefore, we impose a nested grid over the input space:

▶ **Definition 18** (Nested Grid). *Given a partition $\mathcal{W}$, let grid Δ_i, corresponding to $\mathcal{W}_i \in \mathcal{W}$, be a set of cells over the input space with length $\frac{(1+\epsilon)^{i+1}}{\epsilon}$. Then a Nested Grid, denoted by Δ, is defined to be $\bigcup_{i\in[t]} \Delta_i$.*

We then randomly shift the nested grid such that at most an ϵ-fraction of intervals in the i^{th} length class lie within a distance $(1+\epsilon)^{i+1}$ of the i^{th} grid. Since this holds for all $\mathcal{W}_i$, and $\mathcal{W}_i$ are a partition of our input, we lose at most an ϵ-fraction of α. We then define the following object that enables us to obtain accurate estimates for each length class.

▶ **Definition 19** (r_i-Structure). *We define an r_i-Structure to be a subset of the Nested Grid, Δ, such that there exists an interval at the i^{th} level of the structure, there exist no intervals in the structure at any level $i' > i$ and all the intervals in the structure at levels $i' < i$ intersect the interval at the i^{th} level.*

Algorithm 4 Sampling r_i-Structures from $\mathcal{R}_i$.

Input: Given a turnstile stream $\mathcal{P}$ with unit weight arbitrary length intervals, with the length being polynomially bounded, $\epsilon > 0$ and $\delta > 0$, the sampling procedure creates a poly $\left(\frac{\log(n)}{\epsilon}\right)$ size sample of the set $\mathcal{R}_i$.

1. Let Δ_i be the i^{th} level of a randomly shifted *Nested Grid* Δ. Let $\mathcal{R}_i$ be the set of r_i-Structures where the topmost cells lie in Δ_i. Let X_i be the correct guess for the number of non-empty cells in Δ_i up to a constant.
2. Agnostically sample cells from Δ_i with probability $p_i = \max\left(\text{poly}\left(\frac{\log(n)}{\epsilon}\right)\frac{1}{X_i}, 1\right)$. Let S_i be the corresponding substream created.
3. For each cell $c \in \mathcal{S}_i$, let r_i^c be a structure (as defined in 5) with c at the topmost level. Run κ_{max}-*Sparse Recovery* on substream $\mathcal{S}_i$.
4. At the end of the stream, verify that r_i^c is a valid r_i-Structure. Let R_i^s be the set of all such structures.
5. If $X_i > \frac{\epsilon \sum_{i\in[t]} X_i}{t}$, keep R_i^s, else discard it.

Output: $\bigcup_{i\in[t]} R_i^s$.

Let $\mathcal{R}_i$ denote the set of all r_i-Structures at level i. Observe that, taking the union over $i \in [t]$ of $\mathcal{R}_i$ gives a partition of the input. Therefore, estimating $|\mathtt{OPT}_{\mathcal{R}_i}|$ separately and summing up the estimates is a good estimator for α.

Similar to the algorithm in Section 5 a key tool we use is k-Sparse Recovery. Intuitively, we subsample poly$\left(\frac{\log(n)}{\epsilon}\right)$ r_i-Structures from the set $\mathcal{R}_i$ to create a substream $\mathcal{R}_i^s$ and run a $\kappa_{\max}$-Sparse Recovery Algorithm on each substream. At the end of the stream, we get an estimate of $|\mathtt{OPT}_{\mathcal{R}_i}|$ that concentrates. We then add up the estimates across all the levels to form our overall estimate. We formally describe the Level Estimator Algorithm in Algorithm 3, assuming we are given access to a black-box sampling algorithm for sampling an r_i-Structure. We describe how to sample r_i-Structures in *turnstile streams* in Algorithm 4.

Next, we consider the problem of estimating α for arbitrary-length intervals assuming that the space available is at most poly$\left(\frac{W_{max}\log(n)}{\epsilon}\right)$, where W_{max} is an upper bound on the ratio of the max to the min length of an interval. We note that this regime is interesting when W_{max} is sublinear in n. We obtain the following result:

▶ **Theorem 20.** *Let $\mathcal{P}$ be an turnstile stream of unit-weight arbitrary-length intervals s.t. the length is polynomially bounded in n and let $\epsilon \in (0, 1/2)$. Let $\mathcal{W}_{max}$ be an upper bound on the ratio of the max to the min length of intervals in $\mathcal{P}$. Then, there exists an algorithm that outputs an estimator Y s.t. the following guarantees hold:*

1. $\frac{\alpha}{(2+\epsilon)} \leq Y \leq \alpha$ *with probability at least* $2/3$.
2. *The total space used is poly*$\left(\frac{\mathcal{W}_{max}\log(n)}{\epsilon}\right)$.

B Unit Radius Disks in Turnstile Streams

In this section, we state our main result for approximating α and β for unit-radius disks in $\mathbb{R}^2$ that are received in a turnstile stream. Given space constraints, we defer the exposition to the full version.

The main algorithmic result we prove is the following:

▶ **Theorem 21.** *Let $\mathcal{P}$ be a sequence of unit-radius disks that are received as a turnstile stream and let $\epsilon \in (0, 1/2)$. Then, there exists an algorithm that outputs an estimator Y such that with probability at least $9/10$, $\left(\frac{\pi}{8\sqrt{3}} + \epsilon\right)\beta \leq Y \leq \beta$ where α is the cardinality of the largest independent set in $\mathcal{P}$. Further, the total space used is $O\left(\text{poly}\left(\frac{\log n}{\epsilon}\right)\right)$.*

C Insertion-Only Streams

In this section, we state our results for estimating the maximum weighted independent set of intervals in insertion-only streams. Recall, [10] show that $\left(\frac{3}{2} + \epsilon\right)$ is tight for the unweighted case in insertion-only streams. We also show a lower bound for estimating the maximum independent set of disks in insertion-only streams. The lower bound for intervals in [10] shows that $\left(\frac{3}{2} - \epsilon\right)$-approximation requires $\Omega(n)$ space and this naturally extends to disks. We improve this to $2 - \epsilon$, implying a strict separation between intervals and disks for insertion-only streams. Note, this is not yet known to be the case for turnstile streams.

Our theorem for weighted MIS of unit interval in insertion-only streams is as follows:

▶ **Theorem 22.** *Let P be an insertion-only stream of weighted unit intervals s.t. the weights are polynomially bounded in n and let $\epsilon \in (0, 1/2)$. Then, there exists an algorithm that outputs an estimator Y s.t. with probability at least $9/10$ the following guarantees hold:*

1. $\frac{2\beta}{3+\epsilon} \leq Y \leq \beta$.
2. *The total space used is poly* $\left(\frac{\log(n)}{\epsilon}\right)$ *bits.*

Next, we describe a lower bound for estimating α for unit disks in insertion-only streams via a reduction from the communication complexity of the `Indexing` problem, which we use as the starting point. We consider the one-way communication model between two players Alice and Bob and each player has access to private randomness. The randomized communication complexity of `Indexing` is well understood in the two-player one-way communication model.

▶ **Definition 23** (`Indexing`). *Let $I_{n,j}$ denote the communication problem where Alice receives as input a bit vector $x \in \{0, 1\}^n$ and Bob receives an index $j \in [n]$. The objective is for Bob to output x_j under the one-round one-way communication model with error probability at most $1/3$.*

▶ **Theorem 24** (Communication Complexity of $\mathtt{I}_{n,j}$). *The randomized one-round one-way communication complexity of $I_{n,j}$ with error probability at most $1/3$ is $\Omega(n)$.*

We begin with considering the stream of disks $\mathcal{P}$. Let `Alg` be a one-pass insertion-only streaming algorithm that estimates the cardinality of the maximum independent set denoted by α. We then show that `Alg` can be used as a subroutine to solve the communication problem $\mathtt{I}_{n,j}$. Therefore, a lower bound on the communication complexity in turn implies a lower bound on the space complexity of `Alg`. Formally,

▶ **Theorem 25.** *Given a stream of disks $\mathcal{P}$, any randomized one-pass insertion-only streaming algorithm `Alg` which approximates α to within a $(2-\epsilon)$-factor, for any $\epsilon > 0$, with error at most $1/3$, requires $\Omega(n)$ space.*

Proof. We show that any such insertion-only streaming algorithm `Alg` can be used to construct a randomized protocol Π to solve the communication problem. Given her input x, Alice constructs a stream of unit disks and runs `Alg` on the stream. Consider the unit circle around the origin. Divide the half-circle above the x-axis into n equally spaced points, denoted by vectors $p_1, p_2, \ldots, p_n$. For $i \in [n]$, if $x_i = 0$, Alice streams a unit disk centered at p_i. If $x_i = 1$, Alice streams a unit disk centered at $-p_i$. After streaming n disks, Alice communicates the memory state of `Alg` to Bob. Bob uses the message received from Alice as the initial state of the algorithm and continues the stream. Recall, Bob's input only consists of a single index j. Therefore, Bob inserts a unit disk centered at $(1+1/n^2)p_j$.

We first observe that all disks inserted by Alice pairwise intersect. Since all her unit radius disks are centered on the unit circle around the origin, the distance between their center and the origin is 1. Since all the disks contain the origin, they pairwise intersect. Now, let us consider the case where $x_j = 0$. Recall, in this case, Alice inserts the disk centered p_j and Bob inserts the disk centered at $(1+1/n^2)p_j$. The distance between their centers is $1/n^2$ and they clearly intersect. Let us now consider the other disks inserted by Alice, centered at points p_i for $i \neq j$. The distance between their centers is

$$\begin{aligned} ||p_i - (1+1/n^2)p_j||_2^2 &= ||p_i||_2^2 + (1+1/n^2)^2||p_j||_2^2 \pm 2(1+1/n^2)\langle p_i, p_j\rangle \\ &\leq 1 + (1+3/n^2) \pm 2(1+1/n^2)\langle p_i, p_j\rangle \end{aligned} \tag{C.1}$$

where the last inequality follows from $(1+1/n^2)^2 = 1+1/n^4+2/n^2 \leq 1+3/n^2$ for sufficiently large n. Since $i \neq j$, $\langle p_i, p_j\rangle \leq 1-\Theta(1/n)$. Note, $(1+1/n^2)(1-\Theta(1/n)) \leq 1-\Theta(n)$ for sufficiently large n. Substituting this above, we get

$$\begin{aligned} ||p_i - (1+1/n^2)p_j||_2^2 &\leq 1 + (1+3/n^2) \pm 2(1+1/n^2)(1-\Theta(1/n)) \\ &\leq 2 + 3/n^2 \pm 2(1-\Theta(1/n)) \\ &\leq 4 - \Theta(1/n) \end{aligned} \tag{C.2}$$

where the last inequality follows from $\Theta(1/n) \geq 3/n^2$ for sufficiently large n. Therefore, the squared distance between the centers is strictly less 4 and the disks do intersect. As a consequence, all disks pairwise intersect and $\alpha = 1$.

Let us now consider the case where $x_j = 1$. Recall Alice inserts a disk centered at $-p_j$ and Bob inserts a disk centered at $(1+1/n^2)p_j$. The distance between the centers is $(2+1/n^2)$, therefore the two disks do not intersect. Then, α is at least 2. We observe that any $(2-\epsilon)$-approximate algorithm `Alg` can distinguish between these two cases because in the first case `Alg` outputs *at most* 1 and in the second case `Alg` outputs *at least* $1+\epsilon$. Therefore it is a valid protocol for solving $\mathtt{I}_{n,j}$. If `Alg` has error at most $1/3$, the protocol has error at most $1/3$. By Theorem 24, any such protocol requires $\Omega(n)$ communication and in turn `Alg` requires $\Omega(n)$ space. ◀