# Interval Selection in Data Streams: Weighted Intervals and the Insertion-Deletion Setting

**Jacques Dark**
Unaffiliated Researcher, Cambridge, UK

**Adithya Diddapur**
School of Computer Science, University of Bristol, UK

**Christian Konrad**
School of Computer Science, University of Bristol, UK

## Abstract

We study the Interval Selection problem in data streams: Given a stream of $n$ intervals on the line, the objective is to compute a largest possible subset of non-overlapping intervals using $O(|OPT|)$ space, where $|OPT|$ is the size of an optimal solution. Previous work gave a 3/2-approximation for unit-length and a 2-approximation for arbitrary-length intervals [Emek et al., ICALP'12]. We extend this line of work to weighted intervals as well as to insertion-deletion streams. Our results include:

1. When considering weighted intervals, a $(3/2+\epsilon)$-approximation can be achieved for unit intervals, but any constant factor approximation for arbitrary-length intervals requires space $\Omega(n)$.
2. In the insertion-deletion setting where intervals can both be added and deleted, we prove that, even without weights, computing a constant factor approximation for arbitrary-length intervals requires space $\Omega(n)$, whereas in the weighted unit-length intervals case a $(2+\epsilon)$-approximation can be obtained.

Our lower bound results are obtained via reductions to the recently introduced Chained-Index communication problem, further demonstrating the strength of this problem in the context of streaming geometric independent set problems.

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

**Keywords and phrases** Streaming Algorithms, Interval Selection, Weighted Intervals, Insertion-deletion Streams

**Digital Object Identifier** 10.4230/LIPIcs.FSTTCS.2023.24

**Funding** *Jacques Dark*: Part of the work of J.D. was done while at the University of Warwick, supported by an EMEA Microsoft Research PhD studentship and European Research Council grant ERC-2014-CoG 647557.
*Christian Konrad*: Supported by EPSRC New Investigator Award EP/V010611/1.

## 1 Introduction

In this work, we study *streaming algorithms* for the Interval Selection problem. Streaming algorithms are characterized by processing the input data sequentially in a single pass while maintaining only a small sketch of the input data in memory. The objective is to design algorithms that use much less space than the size of the input stream. Due to the low memory requirements, streaming algorithms are well-suited for processing large data sets and constitute a well-established class of resource-constraint algorithms within algorithms research today.

Given a set $\mathcal{S}$ of $n$ intervals on the line, the Interval Selection problem consists of finding a largest subset $T \subseteq \mathcal{S}$ of pairwise non-overlapping intervals and can be regarded as the Maximum Independent Set problem on *interval graphs*, i.e., the intersection graphs of intervals on the line. In the streaming model, we assume that the intervals $\mathcal{S}$ arrive in arbitrary order (for example, in the form of tuples indicating their left and right boundaries). We distinguish between unit-length/arbitrary-length intervals and weighted/unweighted intervals. We will also consider both insertion-only and insertion-deletion streams, where, in the latter, previously inserted intervals may eventually be deleted.

Emek et al. [7] initiated the study of the Interval Selection problem in data streams. They gave a $\frac{3}{2}$-approximation algorithm for unit-length intervals and a 2-approximation algorithm for arbitrary-length intervals, with both algorithms using space $O(|OPT|)$, where $OPT$ denotes an optimal solution (see also [3] for significantly simpler algorithms with the same quality guarantees). As proved by Emek et al., both results are optimal in that computing slightly better approximations requires space $\Omega(n)$. Bakshi et al. [1] were the first to consider weighted intervals and insertion-deletion streams, albeit through a slightly different lens: Their objective was to approximate the total weight of a heaviest independent set of intervals rather than outputting the intervals themselves. They prove that the weight of a heaviest independent set of weighted unit-length intervals in insertion-deletion streams can be approximated within a factor $2 + \epsilon$ with only $\mathrm{poly}(\frac{\log n}{\epsilon})$ space. Curiously, neither weighted intervals nor insertion-deletion streams have previously been considered in the setting where intervals need to be reported.

## 1.1 Our Results

In this work, we address this shortcoming and consider both weighted intervals and insertion-deletion streams. Without loss of generality, if weights are present then we assume that the maximum weight is $W$ and the minimum weight is 1. Similarly, in the arbitrary-length case, we assume that the longest interval is of length $\Delta$ and the shortest interval is of length 1. Other settings can be obtained by scaling. We also mention at this stage that we assume that any interval (weighted or unweighted) can be stored in space $O(1)$ in order to avoid the complexities arising from how interval boundaries and weights are encoded. If instead space $t$ is accounted for storing an interval then the space complexities of our algorithms need to be multiplied by $t$.

We observe that either a particular setting can be solved well in that a constant factor approximation can be achieved with space linear in $|OPT|$, or, a lower bound can be established, demonstrating that space $\Omega(n)$ is necessary for a constant factor approximation in the worst case, i.e., if $W$ or $\Delta$ is large enough.

**Insertion-only Streams.** As our point of departure, we recall that, in insertion-only streams, $O(1)$-approximations for unweighted intervals can be achieved for both unit-length and arbitrary-length intervals with space $O(|OPT|)$ [7]. As our first set of results, we show that, when adding weights to these settings, then the unit-length interval case can still be solved well, however, in the arbitrary-length case, space $\Omega(n)$ is needed in the worst case:

▶ **Theorem 1.** *For every $\epsilon > 0$, there is a deterministic one-pass insertion-only $(\frac{3}{2} + \epsilon)$-approximation streaming algorithm for* Interval Selection *with space $O(\frac{\log(1/\epsilon)}{\epsilon}|OPT|)$ on weighted unit-length intervals.*

▶ **Theorem 2.** *Any one-pass constant error randomized streaming algorithm with approximation factor $\alpha$ for* Interval Selection *on weighted arbitrary-length intervals in insertion-only streams must use $\Omega\left(\frac{1}{\alpha^2}\min\{\Delta^{\frac{1}{2\alpha}}, \frac{n}{2^{2\alpha}}\}\right)$ bits of space.*

We observe that if $\Delta = \mathrm{poly}(n)$ then the previous lower bound shows that space $\Omega(n)$ is needed to achieve a constant factor approximation.

**Insertion-deletion Streams.** We first observe that, in insertion-deletion streams, space as a function of $|OPT|$ cannot be achieved. To this end, consider an input stream consisting of $\Theta(n)$ disjoint intervals that are subsequently either deleted or not, each with probability $\frac{1}{2}$. On such an input distribution, any competitive algorithm must store most of these intervals since the algorithm cannot know whether the intervals will subsequently be deleted. Observe that, if the intervals are indeed deleted, then we have $|OPT| = 0$, while every constant factor approximation algorithm still requires space $\Omega(n)$. To circumvent this issue, we express the space complexities of our insertion-deletion algorithms in terms of $|OPT^*|$, where $OPT^*$ is defined as the largest independent set of intervals *on any prefix of the input stream.* Note that, even in the weighted case, $OPT^*$ is defined as the *largest* independent set on any prefix. While it appears natural to consider the heaviest independent set on any prefix instead, this quantity is not useful since, for example, one could prefix any input stream with the insertion and subsequent deletion of an arbitrarily heavy interval. In this case, the size of a heaviest independent set on any prefix of the stream would always be one.

As our second set of results, we show that unit-length intervals (both unweighted and weighted) can be solved well in insertion-deletion streams, but, unlike in the insertion-only model, constant factor approximations in the unweighted arbitrary-length case require space $\Omega(n)$ in the worst case.

▶ **Theorem 3.** *There is a randomized one-pass insertion-deletion* 2*-approximation streaming algorithm for* Interval Selection *on unweighted unit-length intervals with space* $O(|OPT^*| \cdot \log^3 n)$ *that succeeds with high probability. In the weighted case, a* $(2+\epsilon)$*-approximation can be achieved with space* $O(|OPT^*| \cdot \log^3(n) \cdot \log(W)/\epsilon)$.

A lower bound by Bakshi et al. [1] shows that the result in the previous theorem is best possible in that, even in the unweighted case, every $(2-\epsilon)$-approximation, for any $\epsilon > 0$, requires $\Omega(n)$ space.

▶ **Theorem 4.** *Any one-pass constant error randomized streaming algorithm with approximation factor* $\alpha$ *for* Interval Selection *on unweighted arbitrary-length intervals in insertion-deletion streams must use* $\Omega\left(\frac{1}{\alpha^2}\min\{\Delta^{\frac{1}{\alpha^2}}, \frac{n}{\alpha}\}\right)$ *bits of space.*

## 1.2 Techniques

We will first discuss the ideas behind our algorithms and then address our lower bounds.

**Algorithms.** To obtain our algorithms, we use an idea that goes back to Hochbaum and Mass [12] as it is applied to the Interval Selection problem by Cabello and Pérez-Lantero [3]. Cabello and Pérez-Lantero observed that, if an optimal algorithm for Interval Selection with space $O(1)$ on unit-length intervals that are all contained in the domain $[0, 3)$ can be obtained, then a $\frac{3}{2}$-approximation algorithm with space $O(|OPT|)$ can be achieved without any restrictions on where the intervals are located. We generalize this idea and show that if an $\alpha$-approximation streaming algorithm with space $s$ for unit length intervals that are located within the domain $[0, \gamma)$ exists, for some integer $\gamma$, then a $(\alpha \cdot \gamma/(\gamma-1))$-approximation streaming algorithm with space $O(s \cdot \gamma \cdot |OPT|)$ without any restrictions on where the intervals are located can be obtained in insertion-only streams, and with space $O((s + \log n) \cdot \gamma \cdot |OPT^*|)$ in insertion-deletion streams. Our generalization works for both the unweighted and weighted cases.

We thus focus on designing streaming algorithms that perform well when all intervals in the input stream are contained in a small window $[0, \gamma)$, for some constant integer $\gamma$. In the insertion-only model, we give a $(1+\epsilon)$-approximation algorithm with space $O(\frac{\log(1/\epsilon)}{\epsilon})$ for the domain $[0, 3)$ for weighted unit-length intervals that makes use of weight classes, which, using the idea mentioned above, translates to a $(\frac{3}{2} + \epsilon)$-approximation algorithms with space $O(\frac{\log(1/\epsilon)}{\epsilon} \cdot |OPT|)$ for intervals without restrictions on the domain. In the insertion-deletion setting, we give a $(1 + \epsilon)$-approximation streaming algorithm with space $O(\frac{\log W}{\epsilon} \cdot \log^3 n)$ for weighted unit-length intervals contained in the domain $[0, 2)$ that relies on $\ell_0$-sampling and the use of weight classes, which translates to a $(2 + \epsilon)$-approximation for arbitrary input instances with space $O(\frac{\log W}{\epsilon} \cdot \log^3 n \cdot |OPT^*|)$.

**Lower Bounds.** Our lower bounds are reductions to the recently introduced one-way $p$-party communication problem $\mathsf{CHAIN}_p(k)$ [5] and to an extension that we introduce here that we denote by $\mathsf{AUG\text{-}CHAIN}_p(k)$.

In the *one-way p-party communication model* (see [14] for an introduction to the topic), $p$ parties, who each hold a portion of the input, communicate in a one-way fashion, i.e., the first party sends a message to the second, who, upon receipt, sends a message to the third party. This continues until party $p$ has received a message from party $p - 1$ and outputs the result of the computation. The objective is to design communication protocols such that the size of the largest message is as small as possible. The parties have access to both private and public randomness, and they are allowed to err with small constant probability. Streaming algorithms give rise to one-way communication protocols, and a lower bound on the size of a largest message in every one-way communication protocol for a problem $\mathsf{P}$ then implies a lower bound on the space required by streaming algorithms for problem $\mathsf{P}$.

The problem $\mathsf{CHAIN}_p(k)$ can be regarded as chaining together $p - 1$ instances of the well-known $\mathsf{INDEX}_k$ problem. $\mathsf{INDEX}_k$ is a two-party communication game where the first party holds a bit string $X = X[0], \dots, X[k-1] \in \{0, 1\}^k$ and the second party holds an index $\sigma \in [k-1] := \{0, \dots, k-1\}$. The first party sends a single message to the second who is tasked with outputting the bit $X[\sigma]$. It is known that randomized constant-error protocols that solve $\mathsf{INDEX}_k$ must send a message of size $\Omega(k)$. In $\mathsf{CHAIN}_p(k)$, each pair of subsequent parties $i, i+1$, for $1 \le i < p$, holds a separate $\mathsf{INDEX}_k$ instance $(X^i, \sigma^i)$. The overall $p - 1$ instances are correlated in that

$$X^1[\sigma^1] = X^2[\sigma^2] = \dots = X^{p-1}[\sigma^{p-1}] = z \in \{0, 1\} ,$$

and the objective for party $p$ is to determine the value of $z$. It is known that a protocol that solves $\mathsf{CHAIN}_p(k)$ needs to send at least one message of size $\Omega(\frac{k}{p^2})$ [8].

The key idea of our lower bound construction for weighted arbitrary-length intervals in the insertion-only model is to encode each $\mathsf{INDEX}_k$ input $(X^i, \sigma^i)$ in $\mathsf{CHAIN}_p(k)$ into one or multiple *clique gadgets*, i.e., a set of $k$ slightly shifted but overlapping potential intervals where the $j$th interval is contained in the gadget if and only if$X^i[j] = 1$. We call the $\sigma^i$th potential interval *special.* The strength of the correlation of the outputs of all the $\mathsf{INDEX}_k$ instances in $\mathsf{CHAIN}_p(k)$ is that either *all* or *none* of the special intervals are contained in the constructed clique gadgets. Our construction is such that, if the special intervals are contained in the instance, then a heavy independent set containing the special intervals exists, and otherwise, no heavy independent set exists. We argue that any streaming algorithm that computes a good enough approximation to Interval Selection can be used as a one-way communication protocol that solves $\mathsf{CHAIN}_p(k)$, which establishes our lower bound.

Our lower bound for the insertion-deletion setting makes use of the $\mathsf{AUG\text{-}CHAIN}_p(k)$ problem that we introduce in this work. $\mathsf{AUG\text{-}CHAIN}_p(k)$ differs from $\mathsf{CHAIN}_p(k)$ in that, instead of chaining $\mathsf{INDEX}_k$ instances together, we chain $\mathsf{AUGMENTED\text{-}INDEX}_k$ instances together. The $\mathsf{AUGMENTED\text{-}INDEX}_k$ problem is a slight modification of $\mathsf{INDEX}_k$ where, besides the index $\sigma$, Bob also holds the prefix $X[0], \dots, X[\sigma - 1]$ of $X$. The lower bound established by Feldman et al. for $\mathsf{CHAIN}_p(k)$ carries over to $\mathsf{AUG\text{-}CHAIN}_p(k)$ essentially without modifications.

The $\mathsf{AUGMENTED\text{-}INDEX}_k$ problem (and thus also the $\mathsf{AUG\text{-}CHAIN}_p(k)$ problem) is well-suited for proving lower bounds in insertion-deletion streams. Since Bob knows the prefix $X[0], \dots, X[\sigma - 1]$, Bob can delete the intervals of the clique gadget introduced by Alice that correspond to the prefix $X[0], \dots, X[\sigma-1]$. This provides an additional advantage over $\mathsf{INDEX}_k$ that allows us to obtain a lower bound for unweighted arbitrary-length intervals that could not be obtained without deletions. For details, please see Section 4.2.

## 1.3 Further Related Work

As previously mentioned, the Interval Selection problem is a special case of the Maximum Independent Set (MIS) problem, which has received significant attention in the streaming model. Despite the fact that MIS is NP-hard and hard to approximate within a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$ [11], in the streaming setting, a one-pass $\alpha$-approximation can be obtained by storing $O(\frac{n^2}{\alpha^2})$ edges, by sampling a random vertex-induced subgraph on $\Theta(\frac{n}{\alpha})$ vertices and computing a maximum independent set in this subgraph in exponential time [10] in the post-processing step. In the vertex-arrival order setting, where vertices arrive one-by-one together with their incident edges that connect to previously arrived vertices, Cormode et al. proved that space $\Omega(n^2/\alpha^6)$ is necessary for an $\alpha$-approximation [5]. Furthermore, efficient streaming algorithms exist for computing independent sets that achieve the Caro-Wei bound, see [9] and [4].

The $\mathsf{CHAIN}_p(k)$ problem was introduced by Cormode et al. [5] in order to establish the lower bound mentioned above for vertex-arrival MIS. They also used $\mathsf{CHAIN}_3(k)$ to show that obtaining a better than 2.5-approximation streaming algorithm to the problem of computing a largest subset of a stream of non-overlapping axis-aligned unit squares requires space $\Omega(n)$. Bhore et al. [2] then also used $\mathsf{CHAIN}_p(k)$ to prove that any constant factor approximation algorithm to solve the interval selection problem on split intervals requires space $\Omega(n)$. Feldman et al. [8] used $\mathsf{CHAIN}_p(k)$ for proving lower bounds in the context of one-way communication protocols for sublinear maximization and gave the strongest known lower bound of $\Omega(n/p^2)$ on the size of at least one message in a protocol that solves the problem.

## 1.4 Outline

In Section 2, we discuss tools and known results that we make use of in the work. Then, in Section 3, we give our results that apply to the insertion-only setting, i.e., our algorithm for weighted unit-length intervals and our lower bound for weighted arbitrary-length intervals. Then, in Section 4, we give our results that address the insertion-deletion setting, namely, our algorithm for weighted unit-length intervals and our lower bound for unweighted arbitrary-length intervals. Finally, we conclude with open problems in Section 5.

## 2 Preliminaries

For an integer $k$, we define $[k] := \{0, 1, \dots, k\}$.

**Interval Selection.** In the Interval Selection problem, we are given a set $\mathcal{S}$ of $n$ (potentially weighted) intervals on the line. If weights are present then, for an interval $I \in \mathcal{S}$, we write $w(I)$ to denote its weight. In the presence of weights, the objective is to identify a subset of pairwise non-overlapping intervals of maximum weight. We denote the maximum interval weight in an instance by $W$ and assume that the smallest weight is 1. If no weights are present then the objective is to find a largest cardinality subset of non-overlapping intervals. We distinguish between unit-length intervals, where all intervals are of length 1, and arbitrary-length intervals. In the arbitrary-length case, we denote by $\Delta$ the length of the longest interval and assume that the shortest interval is of length 1.

Let $OPT$ denote an independent set of intervals of maximum cardinality (in the unweighted case) or of maximum weight (in the weighted case) at the end of the stream. For insertion-deletion streams, we will also use $OPT^*$, which denotes an independent set of intervals of largest cardinality across all prefixes of the stream. Then, we denote the cardinality of these solutions by $|OPT|$ ($|OPT^*|$, respectively), and the weight by $w(OPT)$ ($w(OPT^*)$, respectively).

**Streaming Models.** In the insertion-only model, a streaming algorithm sees the $n$ intervals in arbitrary order. In the insertion-deletion model, the input stream consists of a sequence of interval insertions and deletion such that, overall, at most $n$ different intervals have been inserted. We assume that only intervals that have previously been inserted can be deleted.

We assume that storing an interval together with its left and right boundaries as well as its weight requires space $O(1)$. This assumption is for simplicity since the focus of our work is not on how to best encode intervals. We remark, however, that if space $t$ is accounted for storing an interval then the space complexities of our algorithms increase by a factor of $t$.

An important technique for designing insertion-deletion streaming algorithms is $\ell_0$-sampling (see, for example, [6]) that we will also use in this work.

**$\ell_0$-Sampling.** Let $a \in \mathbb{N}_0^n$ be a non-zero vector. The objective in $\ell_0$-sampling is to sample a uniform random non-zero coordinate from $a$. An insertion-deletion stream can be described as a sequence of increases/decreases of the coordinates of an underlying $n$-dimensional vector. Applied to the Interval Selection problem, each coordinate corresponds to an interval in the input stream.

In this work we use the $\ell_0$-sampler by Jowhari et al. [13].

▶ **Theorem 5** ([13]). *There exists an insertion-deletion streaming algorithm that performs $\ell_0$-sampling which uses space $O(\log^2 n \log(1/\delta))$ and succeeds with probability at least $1 - \delta$.*

**Communication Problems.** In this paper, we work with the $\mathsf{CHAIN_p(k)}$ problem introduced by Cormode et al. [5].

▶ **Definition 6.** *$\mathsf{CHAIN_p(k)}$ is a one-way $p$-party communication problem where the players hold parts of the input as follows:*

- *The first player $p_1$ holds a vector $X^1 \in \{0,1\}^k$.*
- *For each $2 \le i \le (p-1)$, player $p_i$ holds a vector $X^i \in \{0,1\}^k$ and an index $\sigma^{i-1} \in [k-1]$.*
- *The last player $p_p$ holds an index $\sigma^{p-1} \in [k-1]$.*

*We are guaranteed that there exists an answer bit $z \in \{0,1\}$ such that $X^i[\sigma^i] = z$ for all $1 \le i \le p-1$. It will be convenient to denote a specific instance of $\mathsf{CHAIN_p(k)}$ by $(X^1, \sigma^1, \ldots, X^{p-1}, \sigma^{p-1})$.*

*The game proceeds with the players sequentially taking turns to send a single message to the next player i.e. $p_1$ sends a message $m^1$ to $p_2$ followed by $p_2$ sending $m^2$ to $p_3$ and so on. The objective of the game is for $p_p$ to output $z$.*

$\mathsf{CHAIN_p(k)}$ can be trivially solved with $O(k)$ bits of total communication: Any of the parties $p_i$ sends $X^i$ to party $p_{i+1}$. Feldman et al. [8] proved the following lower bound:

▶ **Theorem 7** ([8]). *For any positive integers $k$ and $p \geq 2$, any (potentially randomized) protocol for $\mathsf{CHAIN_p(k)}$ with success probability at least $2/3$ must send a message of size $\Omega(k/p^2)$.*

We now define the related multi-party one-way communication problem $\mathsf{AUG\text{-}CHAIN_p(k)}$.

▶ **Definition 8.** *$\mathsf{AUG\text{-}CHAIN_p(k)}$ is a $p$-party one-way communication problem where:*
- *The first player holds a vector $X^1 \in \{0,1\}^k$.*
- *For each $2 \leq i \leq (p-1)$, $p_i$ holds a vector $X^i \in \{0,1\}^k$, an index $\sigma^{i-1} \in [k-1]$, and the prefix $X_+^{i-1} = (X^{i-1}[0], \dots, X^{i-1}[\sigma^{i-1}-1])$.*
- *The last player $p_p$ holds an index $\sigma^{p-1} \in [k-1]$ and the prefix $X_+^{p-1} = (X^{p-1}[0], \dots, X^{p-1}[\sigma^{p-1}-1])$.*

*Again we are guaranteed that there exists an answer bit $z \in \{0,1\}$ such that $X^i[\sigma^i] = z$ for all $1 \leq i \leq p-1$, and we denote a specific instance by $(X^1, \sigma^1, \dots, X^{p-1}, \sigma^{p-1})$. The game then proceeds the same as for $\mathsf{CHAIN_p(k)}$.*

It is not hard to see that the analysis by Feldman et al. for $\mathsf{CHAIN_p(k)}$ also holds for $\mathsf{AUG\text{-}CHAIN_p(k)}$ with essentially no modification. We thus have:

▶ **Theorem 9.** *For any positive integers $k$ and $p \geq 2$, any (potentially randomized) protocol for $\mathsf{AUG\text{-}CHAIN_p(k)}$ with success probability at least $2/3$ must send a message of size at least $\Omega(k/p^2)$.*

## 3 Insertion-only Streams

In this section, we give our $(\frac{3}{2}+\epsilon)$-approximation streaming algorithm for weighted unit-length intervals. To this end, we first show in Subsection 3.1 that if an algorithm can be designed for unit intervals that arrive in the restricted domain $[0,\gamma)$, for some integer $\gamma$, then an algorithm with slightly increased approximation guarantee can be obtained without any restrictions on the domain of the intervals. Then, we present our algorithm in Subsection 3.2.

### 3.1 From Small Domains to Unrestricted Domains

We begin with notation for partitioning the real line into windows of integral length $\gamma$. We define a single window of length $\gamma$ located at $i \in \mathbb{R}$ as $W_i^\gamma = [i, i+\gamma)$. We then define a partition of the real line as $\mathbb{W}_i^\gamma = \left\{W_{i+j\cdot\gamma}^\gamma : j \in \mathbb{Z}\right\}$.

▶ **Lemma 10.** *Let $\mathcal{A}$ be a one-pass insertion-only (insertion-deletion) streaming algorithm with approximation factor $\alpha$ and space $O(s)$ on unit-length weighted intervals such that all intervals are fully contained within the restricted domain $W_0^\gamma = [0,\gamma)$, for some integer $\gamma \geq 2$. Then there exists a one-pass insertion-only (respectively insertion-deletion) streaming algorithm $\mathcal{A}'$ with approximation factor $\alpha \cdot \gamma/(\gamma-1)$ and space $O(s \cdot \gamma \cdot |OPT|)$ (respectively $O((s + \log n) \cdot \gamma \cdot |OPT^*|)$) for unit interval streams with unrestricted interval domains.*

**Proof.** To extend $\mathcal{A}$ to $\mathcal{A}'$ we make use of $\gamma$ partitions of the real line $\mathbb{W}_0^\gamma, \dots, \mathbb{W}_{\gamma-1}^\gamma$. Let $I$ be an interval which arrives in the stream and let $\mathcal{W}$ be the set of windows across all partitions which fully contain $I$. Then, for each $w \in \mathcal{W}$, if $w$ does not fully contain any existing intervals at that point in the stream then $\mathcal{A}'$ creates a new instance of $\mathcal{A}$ in $w$. Then, for each $w \in \mathcal{W}$, $I$ is passed as an input to the instance of $\mathcal{A}$ associated with $w$, translated so that it appears to be contained in $W_0^\gamma$ from the perspective of the instance of $\mathcal{A}$. We denote a window as active if it contains at least one interval and therefore has a running instance of $\mathcal{A}$. In the insertion-deletion setting, each active window also maintains a counter of the number of intervals that it currently contains, and if this counter drops to 0 then $\mathcal{A}'$ clears the corresponding instance of $\mathcal{A}$ and marks the window as inactive. Since the windows within a partition $\mathbb{W}_a^\gamma$ are pairwise disjoint, taking the union of independent sets of intervals across the windows in a partition will also yield an independent set, which we denote as $ALG_a$. At the end of the stream, $\mathcal{A}'$ outputs the largest $ALG_a$ across $a \in \{0, \ldots, \gamma - 1\}$.

**Approximation Factor.** We begin with the approximation factor of $\mathcal{A}'$. Let $OPT$ be a heaviest independent set, and for each $0 \leq a \leq \gamma - 1$, let $OPT_a$ be the set of intervals in $OPT$ which are fully contained in the windows of $\mathbb{W}_a^\gamma$.

First, observe that every interval is contained in precisely $\gamma - 1$ windows. Thus, applying this observation to $OPT$, we obtain

$$\sum_{a=0}^{\gamma-1} w(OPT_a) = (\gamma - 1) w(OPT) \ .$$

Let $j$ be such that $w(OPT_j) = \max\{w(OPT_a) : a \in \{0, \ldots, \gamma-1\}\}$. Then, from the previous equation, we obtain

$$w(OPT) \leq \frac{\gamma}{\gamma - 1} \cdot w(OPT_j).$$

Next, since $\mathcal{A}$ returns an $\alpha$-approximation on each active window and they are disjoint within a given partition $\mathbb{W}_j^\gamma$, we get that $w(OPT_j) \leq \alpha \cdot w(ALG_j)$, and so

$$w(OPT) \leq \frac{\gamma}{\gamma - 1} w(OPT_j) \leq \frac{\gamma}{\gamma - 1} \cdot \alpha \cdot w(ALG_j) \ ,$$

which establishes the approximation factor of $\mathcal{A}'$.

**Space Requirements.** We first argue the space bound in the insertion-only setting. As previously mentioned, every interval is contained in exactly $\gamma - 1$ windows across all the window partitions. Hence, the intervals of $OPT$ are contained in at most $(\gamma - 1) \cdot |OPT|$ different windows. Denote this set of windows by $\mathcal{W}_{OPT}$. Then, based on $\mathcal{W}_{OPT}$, we define the set of windows $\mathcal{W}' \supset \mathcal{W}_{OPT}$ as follows: For each window $[a, a+\gamma) \in \mathcal{W}_{OPT}$, add the two windows $[a-1, a-1+\gamma)$ and $[a+1, a+1+\gamma)$ to $\mathcal{W}'$. Then, observe that $|\mathcal{W}'| \leq 3 \cdot |\mathcal{W}|$.

We will now argue that, for every interval $I$ in the input stream, the $\gamma - 1$ windows that $I$ is contained in are also in $\mathcal{W}'$. Indeed, observe that, since $OPT$ is also a maximal independent set, $I$ intersects with an interval $J \in OPT$, and assume that $I$ intersects $J$ on $J$'s left boundary (the case where $I$ intersects on the right boundary is similar and omitted). Consider now all the $\gamma - 1$ windows that contain $J$. Then, for each of these windows, either $I$ is also contained in it or it is contained in the window shifted by 1 to the left. Since we added these shifted windows to $\mathcal{W}'$, it follows that all of the $\gamma - 1$ windows that contain $I$ are thus also contained in $\mathcal{W}'$. Hence, $\mathcal{A}'$ executes at most $|\mathcal{W}'| \leq 3 \cdot (\gamma - 1) \cdot |OPT|$ instances of $\mathcal{A}$, and since each instance requires space $s$, we conclude that $\mathcal{A}$ uses space $O(s \cdot \gamma \cdot |OPT|)$, which establishes the space bound of the algorithm in the insertion-only setting.

Consider now the insertion-deletion setting. Consider any moment during the processing of the stream, and let $OPT'$ be an independent set of maximum cardinality at this moment. For the exact same reasoning as above, there are $O(\gamma \cdot |OPT'|)$ instances of $\mathcal{A}$ simultaneously running at this moment. Observe that, in addition to the memory state of $\mathcal{A}$, for each simultaneous run, the algorithm $\mathcal{A}'$ also maintains a counter of the number of intervals that currently exist in the respective window. The space complexity in the insertion-deletion setting is thus $O(\gamma \cdot |OPT'| \cdot (s + \log n))$. The result follows since $|OPT'| \leq |OPT^*|$ holds. ◀

## 3.2 Algorithm for Weighted Unit-length Intervals

Equipped with Lemma 10, we are now ready to prove the following result:

▶ **Theorem 1.** *For every $\epsilon > 0$, there is a deterministic one-pass insertion-only $(\frac{3}{2} + \epsilon)$-approximation streaming algorithm for Interval Selection with space $O(\frac{\log(1/\epsilon)}{\epsilon}|OPT|)$ on weighted unit-length intervals.*

Theorem 1 is an immediate consequence of Lemma 10 and Algorithm 1, which is analyzed in Lemma 11.

Algorithm 1 is a $(1 + \epsilon)$-approximation algorithm for the weighted unit-length case when all intervals are included in the domain $[0, 3)$. The algorithm maintains the heaviest interval $I_{\max}$ observed so far and would only retain further intervals of weight at least $\epsilon \cdot w(I_{\max})$. The algorithm uses exponentially increasing weight classes $[(1 + \epsilon)^i, (1 + \epsilon)^{i+1})$, and for each weight class, stores the leftmost and rightmost intervals that fall within the respective weight class, where leftmost and rightmost are defined in the obvious way. We prove in Lemma 11 that the heaviest independent set that can be formed from the intervals stored constitutes a $(1 + \epsilon)$-approximation.

**Algorithm 1** $(1 + \epsilon)$-approximate one-pass streaming algorithm for weighted unit-length intervals on the restricted domain $[0, 3)$.

```
Require: An insertion-only stream of weighted intervals contained within the window $[0, 3)$
 1: $I_{\max} \leftarrow \perp$ (we assume that $w(\perp) = -\infty$){Maintain the heaviest interval}
 2: for each interval $I$ in the input stream do
 3:    if $w(I) > w(I_{\max})$ then
 4:       $I_{\max} \leftarrow I$
 5:       remove all intervals stored with weight less than $\epsilon \cdot w(I_{\max})$
 6:    else if $w(I) < \epsilon \cdot w(I_{\max})$ then
 7:       continue with next interval in stream
 8:    end if
 9:    Let $j$ be the weight class of $I$, i.e., let $j$ be such that $(1 + \epsilon)^{j-1} \leq w(I) < (1 + \epsilon)^j$
10:    Let $S_j$ be the set of (at most two) intervals stored from weight class $j$
11:    $S_j \leftarrow$ left-most and right-most intervals among $S_j \cup \{I\}$
12: end for
13: return Heaviest independent set among all intervals stored
```

▶ **Lemma 11.** *Algorithm 1 is a deterministic one-pass $(1 + \epsilon)$-approximation streaming algorithm for Interval Selection with space $O(\frac{\log(1/\epsilon)}{\epsilon})$ on weighted unit-length intervals that are restricted to the domain $[0, 3)$.*

**Proof.** We first establish the space required by the algorithm. Consider any moment during the processing of the stream and let $I_{\max}$ be the heaviest interval. Let $j^*$ denote the weight class that contains $I_{\max}$, i.e., $j^*$ is such that $(1+\epsilon)^{j^*-1} \leq w(I_{\max}) < (1+\epsilon)^{j^*}$. The algorithm only stores intervals of weight at least $\epsilon \cdot w(I_{\max})$, and at most two intervals per weight class. Recall that the maximum weight in weight class $j$ is $(1 + \epsilon)^j$. Then, if $(1 + \epsilon)^j < \epsilon(1 + \epsilon)^{j^*-1}$ then intervals in weight class $j$ are not stored. This implies that, if intervals in a weight class $j$ are stored, then

$$(1+\epsilon)^j \geq \epsilon(1+\epsilon)^{j^*-1} \text{ , which implies}$$
$$j^* - j = O\left(\frac{\log(1/\epsilon)}{\epsilon}\right) .$$

Hence, there are only $O\left(\frac{\log(1/\epsilon)}{\epsilon}\right)$ weight classes where intervals are stored. This establishes the space complexity of the algorithm.

Regarding the approximation factor, let $OPT$ denote an optimal solution. If $|OPT| = 1$ then our algorithm finds an optimal solution since the algorithm stores the heaviest interval. Hence, assume that $|OPT| = 2$. Suppose first that at least one of the two intervals is of weight at most $\epsilon \cdot w(I_{\max})$. Then, the interval $I_{\max}$ yields a $(1+\epsilon)$-approximation. We can thus assume that both intervals in $OPT$ are of weight at least $\epsilon \cdot w(I_{\max})$. Let $OPT = \{I_L, I_R\}$, and assume that $I_L$ is located to the left of $I_R$. Furthermore, let $j_L$ denote the weight class of $I_L$ and $j_R$ denote the weight class of $I_R$. Then, let $I'_L$ be the left-most interval stored by the algorithm in weight class $j_L$, and let $I'_R$ be the right-most interval stored in weight class $j_R$. Observe that these intervals exist since $I_L$ and $I_R$ also arrived in the stream, and the algorithm either stores these intervals or different intervals of these weight classes that are located further to the left of $I_L$ or to the right of $I_R$. Since $I_L$ and $I'_L$ are in the same weight class, and since $I_R$ and $I'_R$ are in the same weight class, we obtain:

$$w(OPT) = w(I_L) + w(I_R) \leq (1+\epsilon)(w(I'_L) + w(I'_R)) .$$

This completes the proof. ◀

We remark that the space required by our algorithm is independent of $W$, the maximum weight of an interval.

Our main theorem of this section then follows from plugging in the algorithm from Lemma 11 into Lemma 10.

## 3.3 Lower Bound for Weighted Arbitrary-length Intervals

In this section, we will prove our lower bound for weighted arbitrary-length intervals in insertion-only streams.

▶ **Theorem 2.** *Any one-pass constant error randomized streaming algorithm with approximation factor $\alpha$ for* Interval Selection *on weighted arbitrary-length intervals in insertion-only streams must use $\Omega\left(\frac{1}{\alpha^2}\min\{\Delta^{\frac{1}{2\alpha}}, \frac{n}{2^{2\alpha}}\}\right)$ bits of space.*

Our construction relies on clique gadgets that we will define in Subsection 3.3.1. Then, we show that a protocol for $\mathsf{CHAIN}_p(k)$ can be obtained from an algorithm for Interval Selection in Subsection 3.3.2. Finally, we give the analysis in Subsection 3.3.3.

### 3.3.1 Clique Gadgets

We now give our definition of clique gadgets and provide an illustration in Figure 1.

▶ **Definition 12** (Clique Gadget). *Let $X \in \{0,1\}^k$ be a binary vector, let $[a,b] \subset \mathbb{R}$ be an interval with $a < b$, and define the* offset $s_{a,b} = \frac{b-a}{2(k+1)}$. *Then, we define an* unweighted clique gadget *with $k$ intervals as*

$$\mathcal{K}(X,k,[a,b]) = \{I_i : 0 \leq i < k, X[i] = 1\} \text{ , where}$$
$$I_i = \left[a + (i+2)\cdot s_{a,b}, \frac{a+b}{2} + i \cdot s_{a,b}\right] .$$

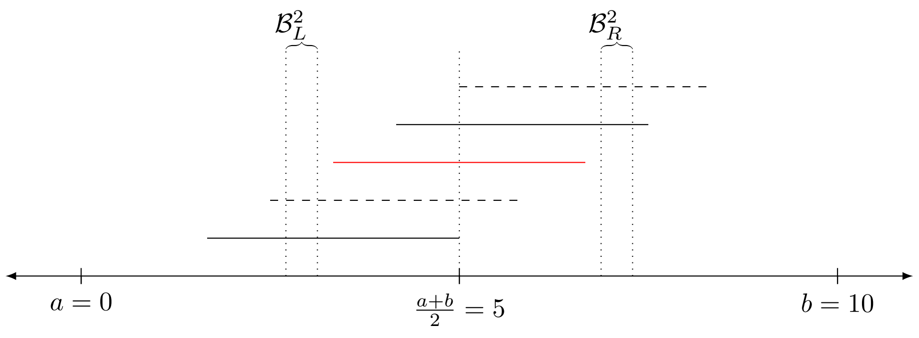

**Figure 1** Example clique gadget $\mathcal{K}(X, k, [a, b])$ for $X = (1, 0, 1, 1, 0)$, $k = 5$, $[a, b] = [0, 10]$, with interval 2 shown in red. The corresponding left and right border regions $\mathcal{B}_L^2$ and $\mathcal{B}_R^2$ are also highlighted noting $|\mathcal{B}_L^2| = |\mathcal{B}_R^2| = s_{a,b}/2 = 5/12$.

*We refer to the intervals $I_i$ as* potential intervals*, and a potential interval $I_i$ is also an actual interval if $X[i] = 1$. For each $0 \le i < k$, we define the left and right border regions of $I_i$ by*

$$\mathcal{B}_L^i = \left[a + \left(i + \frac{5}{4}\right) \cdot s_{a,b}, a + \left(i + \frac{7}{4}\right) \cdot s_{a,b}\right] ,$$
$$\mathcal{B}_R^i = \left[\frac{a+b}{2} + \left(i + \frac{1}{4}\right) \cdot s_{a,b}, \frac{a+b}{2} + \left(i + \frac{3}{4}\right) \cdot s_{a,b}\right] ,$$

*Last, we define a weighted clique gadget $\mathcal{K}(X, k, [a, b], w)$ as the unweighted clique gadget $\mathcal{K}(X, k, [a, b])$ where each interval has weight $w \in \mathbb{R}$.*

Clique gadgets have the following properties:

▶ **Observation 13.** *The clique gadget $\mathcal{K}(X, k, [a, b])$ has the following properties:*

1. *All intervals and border regions are fully contained within the set $(a, b)$.*
2. *All intervals contain the midpoint $m = \frac{b+a}{2}$ and thus form a clique.*
3. *For every $i$, the border regions have width $|\mathcal{B}_L^i| = |\mathcal{B}_R^i| = s_{a,b}/2 > 0$.*
4. *For every $i$, every point in $\mathcal{B}_L^i$ intersects with every potential interval $I_j$ where $j < i$.*
5. *For every $i$, every point in $\mathcal{B}_R^i$ intersects with every potential interval $I_j$ where $i < j$.*
6. *For every $i$, $(\mathcal{B}_L^i \cup \mathcal{B}_R^i) \cap I_i = \emptyset$.*

### 3.3.2 Reduction

We prove our lower bound via a reduction to the $\mathsf{CHAIN}_p(k)$ problem. To this end, we will show that a one-pass streaming algorithm $\mathcal{A}$ for Interval Selection for weighted arbitrary-length intervals can be used to obtain a protocol $\pi_{\text{Chain}}$ that solves $\mathsf{CHAIN}_p(k)$.

The reduction is described in Figure 2.

**Construction of $\pi_{\textbf{Chain}}$:**

For parameters $p$ and $k$, let $(X^1, \sigma^1, \dots, X^{p-1}, \sigma^{p-1})$ be a $\mathsf{CHAIN}_p(k)$ instance.

The parties base their construction on positive reals $a, b, w$ with $a < b$ whose values we determine later. The parties proceed as follows:

- **Party 1:** Based on input $X^1$, party 1 constructs the clique gadget $\mathcal{K}(X^1, k, [a, b], w)$, runs $\mathcal{A}$ on the created intervals, and send the memory state of $\mathcal{A}$ to party 2.
- **Party $1 < i \le p$:** Denote by $\mathcal{K}_1, \dots, \mathcal{K}_{2^{i-2}}$ the $2^{i-2}$ clique gadgets introduced by party $i-1$, and let $B_{L,j}$ and $B_{R,j}$ denote the left and right border regions of the $\sigma^{i-1}$th potential interval in $\mathcal{K}_j$. For each $j \in \{1, \dots, 2^{i-2}\}$, if $i < p$ then party $i$ introduces the clique gadgets $\mathcal{K}(X^i, k, B_{L,j}, \frac{w}{2^{i-1}})$ and $\mathcal{K}(X^i, k, B_{R,j}, \frac{w}{2^{i-1}})$, and if $i = p$ then party $i = p$ introduces two arbitrary intervals of weight $\frac{w}{2^{p-1}}$ that are fully contained in $B_{L,j}$ and $B_{R,j}$, recalling there is no vector $X^p$ in a $\mathsf{CHAIN}_p(k)$ instance.

  Party $i$ receives the memory state of an execution of $\mathcal{A}$ from party $i-1$, and then continues the execution of $\mathcal{A}$ on the intervals created, and if $i < p$, forwards the memory state to the next party. If $i = p$, then, as we will detail further below, party $p$ is able to solve $\mathsf{CHAIN}_p(k)$ based on the output of $\mathcal{A}$.

**Figure 2** Construction of $\pi_{\text{Chain}}$ for solving $\mathsf{CHAIN}_p(k)$ given an algorithm $\mathcal{A}$ for Interval Selection on weighted unit-length intervals.

We observe that the clique gadgets created by the $p$ parties can be represented as a rooted binary tree $\mathcal{T}$, where the first party's gadget constitutes the root node. For party $1 < i < p$, let $\mathcal{K}$ be a gadget inserted by party $i-1$, $\mathcal{K}_L$ and $\mathcal{K}_R$ the two gadgets inserted by party $i$ into the left and right border regions of the special interval in $\mathcal{K}$. Then $\mathcal{K}_L$ and $\mathcal{K}_R$ are the two children of $\mathcal{K}$ in $\mathcal{T}$. The last party's intervals constitute the leaves of the tree. We will use $\mathcal{T}$ to determine the sizes of maximum independent sets when the underlying $\mathsf{CHAIN}_p(k)$ instance evaluates to either 1 or 0.

In this reduction, party $i$ introduces $2^{i-1}$ clique gadgets and therefore at most $k \cdot 2^{i-1}$ intervals. The total number of intervals $n$ is thus bounded by $n \le \sum_{i=1}^{p} k \cdot 2^{i-1} < k \cdot 2^p$.

### 3.3.3 Analysis of the Reduction

We will argue in Lemma 14 that, depending on whether the answer bit in the $\mathsf{CHAIN}_p(k)$ instance is 1 or 0, the weight of a heaviest independent set is very different. Then, we conclude the section with the proof of our main result, Theorem 2.

▶ **Lemma 14.** *If the answer bit in the $\mathsf{CHAIN}_p(k)$ instance is* 1 *then the Interval Selection instance constructed in the reduction has an independent set of weight $w \cdot p$. If the answer bit is* 0 *then every independent set has weight at most $2 \cdot w$.*

**Proof.** The properties we use here to determine the independence of intervals from different gadgets follow from Observation 13.

Suppose first that the answer bit is 1. Then, for each party $1 \le i < p$ and for each clique gadget constructed by party $i$, we pick the $\sigma^i$th interval of the gadget and add it to an independent set. We also pick all intervals created by party $p$. Observe that the resulting independent set corresponds to picking one interval from each node in $\mathcal{T}$. Overall, this creates an independent set of weight

$$\sum_{i=1}^{p} \frac{2^{i-1} \cdot w}{2^{i-1}} = w \cdot p \ ,$$

since party $i$ holds $2^{i-1}$ clique gadgets where each interval has weight $\frac{w}{2^{i-1}}$.

Suppose now that the answer bit is 0. We argue via the tree structure $\mathcal{T}$ that every independent set is of weight at most $2 \cdot w$. To this end, we observe that, by construction of the instance, if an interval $I$ from a clique gadget $\mathcal{K}$ is selected then $I$ overlaps with all intervals in either the left or the right subtree of $\mathcal{K}$. Hence, since $I$ is selected, none of the intervals in either the left or the right subtree of $\mathcal{K}$ can also be selected.

Consider now any independent set of intervals $\mathcal{I}$, and let $\mathcal{I}'$ be the independent set consisting only of party $p$'s intervals. We will now argue that

$$w(\mathcal{I}) \leq 2 \cdot w(\mathcal{I}') \ , \tag{1}$$

and, since $w(\mathcal{I}') = w$, the result then follows.

To see why Inequality 1 holds, we conduct local transformations to turn $\mathcal{I}$ into $\mathcal{I}'$. To this end, repeat the following process: for each interval $I \in \mathcal{I}$ that is contained in a clique gadget $\mathcal{K}$ held by party $i < p$, we remove $I$ from $\mathcal{I}$. Let $\mathcal{K}'$ be the child of $\mathcal{K}$ that does not contribute any intervals to $\mathcal{I}$. As previously argued, $\mathcal{K}'$ must exist. Then, we add back all of party $p$'s intervals that are contained in $\mathcal{K}'$. When this process terminates then we have transformed $\mathcal{I}$ into $\mathcal{I}'$. Observe that, when removing $I$, we subtracted a weight of $w(I) = \frac{w}{2^{i-1}}$ from $\mathcal{I}$ and subsequently added $2^{p-i-1}$ intervals, each of weight $\frac{w}{2^{p-1}}$, back into $\mathcal{I}$, which corresponds to a total weight of $2^{p-i-1} \cdot \frac{w}{2^{p-1}} = \frac{w}{2^i} = w(I)/2$. Hence, overall, at most half of the initial weight of $\mathcal{I}$ is lost in this transformation, which proves Inequality 1 and completes the proof. ◀

**Proof of Theorem 2.** From Lemma 14, we obtain that if the answer bit in the $\mathsf{CHAIN}_p(k)$ instance is 1 then the instance has an independent set of weight $w \cdot p$, and if the answer bit is 0 then every independent set in the instance is of weight at most $2 \cdot w$. Hence, in our reduction, by the output of $\mathcal{A}$, party $p$ can distinguish the two cases as long as the approximation factor $\alpha$ of $\mathcal{A}$ is such that $\alpha < \frac{p}{2}$. Theorem 7 thus yields that algorithm $\mathcal{A}$ must use space $\Omega(\frac{k}{p^2})$. It remains to pick suitable values for $k$ and $p$. We pick $p = 2\alpha + 1$. We observe that the ratio between the longest and shortest intervals in our construction is $\Delta = (4(k+1))^{p-1} = (4(k+1))^{2\alpha}$, which implies that $k = \frac{1}{4}\Delta^{\frac{1}{2\alpha}} - 1$. Our lower bound thus becomes $\Omega(\Delta^{\frac{1}{2\alpha}}/\alpha^2)$. Since our construction uses $n \leq k \cdot 2^p = (\frac{1}{4}\Delta^{\frac{1}{2\alpha}} - 1) \cdot 2^{2\alpha+1} \leq \Delta^{\frac{1}{2\alpha}} \cdot 2^{2\alpha}$ intervals, our lower bound can never exceed the bound $O(\frac{n}{2^{2\alpha} \cdot \alpha^2})$.

Last, the construction departs with arbitrary reals $a, b, w$ that determine the domain of the construction and the heaviest weight. We pick any $a$ and select $b$ large enough such that the shortest interval is of length 1, and we pick $w$ large enough such that the lightest interval is of weight 1. This completes the proof. ◀

## 4 Insertion-deletion Streams

### 4.1 Algorithm for Unit-length Intervals

In this section, we give our insertion-deletion streaming algorithms for Interval Selection for both weighted and unweighted unit-length intervals. We prove the following theorem:

▶ **Theorem 3.** *There is a randomized one-pass insertion-deletion* $2$*-approximation streaming algorithm for* Interval Selection *on unweighted unit-length intervals with space* $O(|OPT^*| \cdot \log^3 n)$ *that succeeds with high probability. In the weighted case, a* $(2+\epsilon)$*-approximation can be achieved with space* $O(|OPT^*| \cdot \log^3(n) \cdot \log(W)/\epsilon)$.

As in the insertion-only case, we argue in Lemma 15 that suitable algorithms for restricted domains exist and then apply Lemma 10 to obtain algorithms for unrestricted domains.

▶ **Lemma 15.** *There is a randomized one-pass insertion-deletion streaming algorithm with space* $O(\log^3 n)$ *for* Interval Selection *that produces an optimal solution for unweighted unit-length intervals that are restricted to the domain* $[0, 2)$*. For weighted intervals, the approximation factor becomes* $1+\epsilon$ *and the space is* $O(\frac{\log W}{\epsilon} \log^3 n)$.

**Proof.** We first consider the unweighted unit-length case. Since every pair of intervals that are contained in $[0, 2)$ intersect, the task reduces to outputting any interval if there is one. To achieve this task, we can run an $\ell_0$-sampler that succeeds with high probability. To this end, we use the $\ell_0$-sampler of Jowhari et al. (see Theorem 5) that uses space $O(\log^3 n)$ and errs with probability $\frac{1}{n^{10}}$.

Regarding weighted intervals, the task is to report an interval of weight at least $\frac{1}{1+\epsilon}$ times the weight of a heaviest interval. To this end, we run an $\ell_0$-sampler for intervals in every weight class $[(1+\epsilon)^{j-1}, (1+\epsilon)^j)$, for every $j \in \{1, \dots, \log_{1+\epsilon} W\}$. Observe that $\log_{1+\epsilon} W = O(\frac{\log W}{\epsilon})$ for small values of $\epsilon$. The space complexity follows since each $\ell_o$-sampler uses space $O(\log^3 n)$. By a union bound, the error probability of the resulting algorithm is at most $\frac{1}{n^{10}}$ times the number of weight classes that contain at least one interval, and since there are at most $n$ such weight classes, the result follows. ◀

**Proof of Theorem 3.** The proof follows immediately from Lemma 15 and Lemma 10. ◀

## 4.2 Lower Bound for Unweighted Arbitrary-length Intervals

We will now prove our lower bound that applies to the unweighted, arbitrary-length setting in insertion-deletion streams:

▶ **Theorem 4.** *Any one-pass constant error randomized streaming algorithm with approximation factor* $\alpha$ *for* Interval Selection *on unweighted arbitrary-length intervals in insertion-deletion streams must use* $\Omega\left(\frac{1}{\alpha^2} \min\{\Delta^{\frac{1}{\alpha^2}}, \frac{n}{\alpha}\}\right)$ *bits of space.*

Similar to the lower bound of Section 3.3, we will give a reduction, however, instead of reducing to $\mathsf{CHAIN}_p(k)$, we reduce to $\mathsf{AUG\text{-}CHAIN}_p(k)$. Our construction will make use of the clique gadgets defined in Section 3.3. We will give our reduction in Section 4.2.1, however, due to space restrictions, the analysis is postponed to Appendix A.

### 4.2.1 Reduction

We will now show that a one-pass insertion-deletion streaming algorithm $\mathcal{A}$ for Interval Selection for unweighted arbitrary-length intervals can be used to obtain a protocol $\pi_{\text{Aug-Chain}}$ that solves $\mathsf{AUG\text{-}CHAIN}_p(k)$. The reduction is described in Figure 3.

**Construction of $\pi_{\textbf{Aug-Chain}}$:** For parameters $p$ and $k$, let $(X^1, \sigma^1, \dots, X^{p-1}, \sigma^{p-1})$ be a AUG-CHAIN$_p(k)$ instance. The parties base their construction on positive reals $a, b$ with $a < b$ whose values we determine later. The parties proceed as follows:

- **Party 1:** Based on input $X^1$, party 1 constructs the unweighted clique gadget $\mathcal{K}(X^1, k, [a, b])$, runs $\mathcal{A}$ on the created intervals, and send the memory state of $\mathcal{A}$ to party 2.
- **Party $1 < i \leq p$:** Denote by $\mathcal{K}$ the clique gadget constructed by party $i-1$, and let $B_R$ denote the right border region of the $\sigma^{i-1}$th potential interval in $\mathcal{K}$. Party $i$ deletes all actual intervals of $\mathcal{K}$ with index below $\sigma^{i-1}$, which is possible since party $i$ knows the prefix $X_+^{i-1} = \{X^{i-1}[0], \dots, X^{i-1}[\sigma^{i-1} - 1]\}$. Then, if $i < p$, party $i$ introduces the clique gadget $\mathcal{K}(X^i, k, B_R)$, and if $i = p$ then party $i = p$ introduces one arbitrary interval that is fully contained in $B_R$.
  Party $i$ receives the memory state of an execution of $\mathcal{A}$ from party $i-1$, and then continues the execution of $\mathcal{A}$ on the intervals created, and if $i < p$, forwards the memory state to the next party. If $i = p$, then, as we will detail further below, party $p$ is able to solve Aug-CHAIN$_p(k)$ based on the output of $\mathcal{A}$.

**Figure 3** Construction of $\pi_{\text{Aug-Chain}}$ for solving AUG-CHAIN$_p(k)$ given an insertion-deletion streaming algorithm $\mathcal{A}$ for Interval Selection on unweighted unit-length intervals.

We observe that the number of intervals $n$ in this construction is bounded by $n \leq k \cdot p$. For the analysis of this reduction, we refer the reader to Appendix A.

## 5 Conclusion

In this paper, we gave streaming algorithms for Interval Selection on weighted unit-length intervals in the insertion-only and the insertion-deletion settings. We also gave space lower bounds, one for the case of weighted arbitrary-length intervals in the insertion-only setting, and one for the case of unweighted arbitrary-length intervals in the insertion-deletion setting.

Our results demonstrate a jump in complexity when going from the insertion-only to the insertion-deletion setting: While a constant factor approximation for arbitrary-length intervals can be achieved in the insertion-only setting with space $O(|OPT|)$, space $\Omega(n)$ is required in the insertion-deletion setting. We also showed that only when combining arbitrary-length intervals with weights then a lower bound of $\Omega(n)$ can be obtained for constant factor approximations in the insertion-only setting.

We conclude with two open problems.

1. Our $(\frac{3}{2} + \epsilon)$-approximation insertion-only streaming algorithm and our $(2 + \epsilon)$-approximation insertion-deletion streaming algorithm for weighted unit-length intervals have space complexities that depend on $\frac{1}{\epsilon}$. Can we prove any lower bounds that capture these dependencies on $\frac{1}{\epsilon}$?
2. Interval Selection is relatively well understood by now. Can we obtain optimal algorithm for other geometric objects, such as axis-aligned unit squares or unit circles? For example, as previously mentioned, regarding axis-aligned unit squares, it is known that a 3-approximation can be obtained with space $O(|OPT|)$, while a lower bound shows that space $\Omega(n)$ is needed for approximation factors below 2.5 [5]. Can we close this gap?

## References

1 Ainesh Bakshi, Nadiia Chepurko, and David P. Woodruff. Weighted maximum independent set of geometric objects in turnstile streams. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 64:1–64:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.64.

2 Sujoy Bhore, Fabian Klute, and Jelle J. Oostveen. On streaming algorithms for geometric independent set and clique. In Parinya Chalermsook and Bundit Laekhanukit, editors, *Approximation and Online Algorithms – 20th International Workshop, WAOA 2022, Potsdam, Germany, September 8-9, 2022, Proceedings*, volume 13538 of *Lecture Notes in Computer Science*, pages 211–224. Springer, 2022. doi:10.1007/978-3-031-18367-6_11.

3 Sergio Cabello and Pablo Pérez-Lantero. Interval selection in the streaming model. *Theoretical Computer Science*, 702:77–96, 2017. doi:10.1016/j.tcs.2017.08.015.

4 Graham Cormode, Jacques Dark, and Christian Konrad. Approximating the caro-wei bound for independent sets in graph streams. In Jon Lee, Giovanni Rinaldi, and Ali Ridha Mahjoub, editors, *Combinatorial Optimization – 5th International Symposium, ISCO 2018, Marrakesh, Morocco, April 11-13, 2018, Revised Selected Papers*, volume 10856 of *Lecture Notes in Computer Science*, pages 101–114. Springer, 2018. doi:10.1007/978-3-319-96151-4_9.

5 Graham Cormode, Jacques Dark, and Christian Konrad. Independent Sets in Vertex-Arrival Streams. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 45:1–45:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.45.

6 Graham Cormode and Donatella Firmani. A unifying framework for $\ell_0$-sampling algorithms. *Distributed Parallel Databases*, 32(3):315–335, 2014. doi:10.1007/s10619-013-7131-9.

7 Yuval Emek, Magnus M. Halldorsson, and Adi Rosen. Space-Constrained Interval Selection. In *39th International Colloquium on Automata, Languages, and Programming (ICALP) 2012, Warwick, UK*, July 2012.

8 Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1363–1374. ACM, 2020. doi:10.1145/3357713.3384286.

9 Bjarni V. Halldórsson, Magnús M. Halldórsson, Elena Losievskaja, and Mario Szegedy. Streaming algorithms for independent sets in sparse hypergraphs. *Algorithmica*, 76(2):490–501, 2016. doi:10.1007/s00453-015-0051-5.

10 Magnús M. Halldórsson, Xiaoming Sun, Mario Szegedy, and Chengu Wang. Streaming and communication complexity of clique approximation. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, volume 7391 of *Lecture Notes in Computer Science*, pages 449–460. Springer, 2012. doi:10.1007/978-3-642-31594-7_38.

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

12 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *J. ACM*, 32(1):130–136, January 1985. doi:10.1145/2455.214106.

13 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*, PODS '11, pages 49–58, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1989284.1989289.

14 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1996. doi:10.1017/CBO9780511574948.

## A Analysis of the Lower Bound Reduction From Section 4.2.1

As in the insertion-only case, we will first argue that the size of a largest independent set is very different depending on whether the answer bit in AUG-CHAIN$_p(k)$ is 1 or 0.

▶ **Lemma 16.** *If the answer bit in the AUG-CHAIN$_p(k)$ instance is* 1 *then the Interval Selection instance constructed in the reduction has an independent set of size $p$. If the answer bit is* 0 *then every independent set is of size at most* 1*.*

**Proof.** The properties we use here to determine the independence of intervals from different gadgets follow from Observation 13.

Let $\mathcal{K}_i$ denote the clique gadget introduced by party $i$ after the deletions introduced by party $i+1$ have been applied.

Suppose first that the answer bit is 1. Then, the set of intervals $\{I_1, \dots, I_p\}$, where $I_i$ is the $\sigma^i$th interval of $\mathcal{K}_i$, is an independent set of size $p$.

Suppose now that the answer bit is 0. Then, it can be seen that all intervals in the instance intersect with the mid-point of the interval added by party $p$. The intervals thus form a clique, and, thus, every independent set is of size at most 1. ◀

**Proof of Theorem 4.** From Lemma 16, we see that if the algorithm $\mathcal{A}$ has an approximation factor $\alpha \leq p - 1$ then party $p$ can distinguish between the 0 and 1 case in AUG-CHAIN$_p(k)$. Hence, we will set $p = \alpha + 1$.

By Theorem 9, algorithm $\mathcal{A}$ must use space $\Omega(\frac{k}{p^2}) = \Omega(\frac{k}{\alpha^2})$. Similar to the construction in Section 3.3, we have $\Delta = (4(k+1))^{p-1} = (4(k+1))^{\alpha}$, which implies $k = \frac{1}{4}\Delta^{\frac{1}{p-1}} - 1$ and gives a lower bound of $\Omega(\Delta^{\frac{1}{p-1}}/\alpha^2)$. Since the construction uses at most $n \leq k \cdot p = (\frac{1}{4}\Delta^{\frac{1}{p-1}} - 1)(\alpha+1)$ intervals, the lower bound cannot exceed $O(\frac{n}{\alpha^3})$.

Last, the construction is based on reals $a, b$. We pick any $a$ and select $b$ large enough such that the shortest interval is of length 1. This concludes the proof. ◀