Streaming Frequent Items with Timestamps and Detecting Large Neighborhoods in Graph Streams

Christian Konrad
christian.konrad@bristol.ac.uk

Department of Computer Science, University of Bristol, UK

Abstract

Detecting frequent items is a fundamental problem in data streaming research. However, in many applications, besides the frequent items themselves, meta data such as the timestamps of when the frequent items appeared or other application-specific data that "arrives" with the frequent items needs to be reported too.

To this end, we introduce the Neighborhood Detection problem in graph streams, which both accurately models situations such as those stated above, and addresses the fundamental problem of detecting large neighborhoods or stars in graph streams. In Neighborhood Detection, an algorithm receives the edges of a bipartite graph $G = (A, B, E)$ with $|A| = n$ and $|B| = \text{poly}\, n$ in arbitrary order and is given a threshold parameter d. Provided that there is at least one A-node of degree at least d, the objective is to output a node $a \in A$ together with at least $\frac{d}{c}$ of its neighbors, where c is the approximation factor.

We show that in insertion-only streams, there is a one-pass $\tilde{\text{O}}(n+n^{\frac{1}{c}}d)$ space c-approximation streaming algorithm, for integral values of $c \geq 2$. We complement this result with a lower bound, showing that computing a $(c/1.01)$-approximation requires space $\Omega(n/c^2+n^{\frac{1}{c-1}}d/c^2)$, for any integral $c \geq 2$, which renders our algorithm optimal for a large range of settings (up to logarithmic factors). In insertion-deletion (turnstile) streams, we give a one-pass c-approximation algorithm with space $\tilde{\text{O}}(\frac{dn}{c^2})$ (if $c \leq \sqrt{n}$). We also prove that this is best possible up to logarithmic factors.

Our insertion-only algorithm combines degree counts with reservoir sampling, while our insertion-deletion algorithm uses both edge and vertex sampling techniques. Both lower bounds are obtained by defining new multi-party and two-party communication problems, respectively, and proving lower bounds on their communication complexities using information theoretic arguments.

1 Introduction

The *streaming model of computation* addresses the fundamental issue that modern massive data sets are too large to fit into the Random-Access Memory (RAM) of modern computers. Typical examples of such data sets are Internet traffic logs, financial transaction streams, database logs, and massive graphical data sets, such as the Web graph and social network graphs. A data streaming algorithm receives its input piece by piece in a linear fashion and has access to only a sublinear amount of memory. This prevents the algorithm from seeing the input in its entirety at any one moment.

Detecting *heavy hitters* or frequent elements is one of the fundamental problems considered in data streaming research. Given a stream $S = s_1, s_2, \ldots, s_n$ of length n with $s_i \in [m]$, for some integer m, the goal is to identify items in $[m]$ that appear at least ϵn times, for some $\epsilon > 0$. This problem was first solved by Misra and Gries in 1982 [41] and has since been addressed in countless research papers (e.g. [15, 38, 23, 40, 19, 35, 9, 11]), culminating in provably optimal algorithms [10]. However, in many applications, only knowing the frequent items is insufficient, and additional application-specific data is required. For example:

- Given a database log, a heavy hitters algorithm can be used to detect a frequently updated (or queried) value in a database. Users, however, that committed these updates (or queries) cannot be reported by such an algorithm.
- Given a stream of friendship updates in a social network graph, a heavy hitters algorithm can detect nodes of large degree (e.g., an influencer in a social network). Their neighbors (e.g., followers of an influencer), however, cannot be outputted by such an algorithm.
- Given the traffic log of an Internet router logging timestamps, source, and destination IP addresses of forwarded IP packages, Denial-of-Service attacks can be detected by identifying *distinct heavy hitters*, that is, frequent target IP addresses that are requested from many distinct sources [24]. Here, a (distinct) heavy hitters algorithm only reports frequent target IP addresses and thus potential machines that were under attack, however, the timestamps of when these attacks occurred or the various source IP addresses from where the attacks originated remain unknown[1].

Applications such as those mentioned above have in common that items in the input stream arrive together with additional application-specific satellite data. Besides the frequent items themselves, the satellite data of frequent items also needs to be reported.

To this end, we introduce the Neighborhood Detection problem, which accurately models the problems mentioned above and addresses the fundamental task of identifying large neighborhoods or stars in *graph streams* (see Section 1.1):

Problem 1 (Neighborhood Detection)**.** *In Neighborhood Detection(n, d), we are given a bipartite graph $G = (A, B, E)$ with $|A| = n$ and $|B| = \operatorname{poly} n$, a threshold parameter d, and the promise that G contains at least one A-vertex of degree at least d. The goal is to output an A-vertex together with at least d/c of its neighbors, for some $c \geq 1$, where c is the approximation factor.*

The bipartite nature of the problem definition reflects applications such as the ones mentioned above, where items correspond to A-vertices, satellite data corresponds to B-vertices, and the

[1]We remark that in this example knowing the sender IP addresses may unfortunately not always be helpful as sender IP addresses are often spoofed in Denial-of-Service attacks.

appearance of an item in the stream corresponds to the insertion of an edge connecting this item to satellite data, thus forming a stream of the edges of a graph. Observe that this representation allows us to associate multiple items to the same satellite data. The restriction $|B| = \operatorname{poly} n$ is imposed for convenience as it is reasonable and simplifies the complexity bounds of our algorithms.

Neighborhood Detection is closely related to the task of approximating the largest star in a graph stream, which is a fundamental problem that deserves attention in its own right:

Problem 2 (Star Detection)**.** *Given a general graph $G = (V, E)$, the Star Detection problem consists of computing the largest star in G, i.e., determining a node of largest degree together with its neighborhood. A c-approximation algorithm to Star Detection outputs a node together with at least Δ/c of its neighbors, where Δ is the maximum degree in the input graph.*

Star Detection ties in with various works in the streaming literature that address the problem of approximating certain graph structures, such as large independent sets and cliques [27, 16], and large matchings (e.g. [26, 34]). We will show that streaming algorithms for Neighborhood Detection can be used to solve Star Detection.

1.1 Graph Streams

Streaming algorithms for graph problems have been studied for twenty years [29], and a multitude of graph problems such as matchings, independent sets, graph sparsification, spanners, connectivity, and different subgraph counting problems have since been addressed in various graph stream models (see [39] for an excellent survey). The following models are relevant to our work:

1. In *insertion-only* streams, the input consists of a sequence of the edges of a graph in arbitrary order. The objective is to design algorithms that make a single pass over the input and use as little space as possible. Concerning Neighborhood Detection, observe that using space $\tilde{\mathrm{O}}(nd)$[2] the problem can be solved *exactly* by storing the first $\min\{\deg(a), d\}$ edges incident to every A-vertex a. Our objective is therefore to obtain algorithms that use space $o(nd)$.

2. In *insertion-deletion* streams (also known as turnstile or dynamic streams), the input consists of a sequence of edge insertions and deletions, where edges can only be deleted if they have previously been inserted, thereby modeling graphs that undergo change. Using l_0-sampling techniques [17, 18], it is possible to sample $\min\{\deg(a), d\}$ edges incident to every A-vertex a in the insertion-deletion model. This also yields a $\tilde{\mathrm{O}}(nd)$ space algorithm that solves Neighborhood Detection exactly.

While some graph problems are equally hard to solve in insertion-only and in insertion-deletion streams (up to poly-log factors in their space complexity), such as Connectivity [1, 42] and Maximum Independent Set [27], others, such as Maximum Matching [33, 5], require substantially more space in insertion-deletion streams. Our results show that approximation algorithms for Neighborhood Detection require substantially more space if deletions are allowed.

It is known that $\Omega(n)$ is a natural space barrier for many graph problems in insertion-only streams [26, 25, 42]. *Semi-streaming* algorithms [26], i.e., algorithms that use space $\mathrm{O}(n \operatorname{polylog} n)$, have therefore received particular attention. We will see that our results yield a $\mathrm{O}(\log n)$-approximation semi-streaming algorithm for Star Detection in insertion-only streams.

[2] We use $\tilde{\mathrm{O}}, \tilde{\Theta}$ and $\tilde{\Omega}$ to mean O, Θ and Ω, respectively, with log factors suppressed.

1.2 Our Results

In this paper, we give streaming algorithms and space lower bounds for Neighborhood Detection in both insertion-only and insertion-deletion streams.

In insertion-only streams, for integers $c \geq 2$, we give a one-pass c-approximation streaming algorithm with space $\tilde{O}(n + n^{\frac{1}{c}}d)$ that succeeds with high probability[3] (**Theorem 3.2**). This algorithm can also be used to obtain a $O(\log n)$-approximation semi-streaming algorithm for Star Detection (**Corollary 3.3**). We complement this result with a lower bound, showing that space $\Omega(n/c^2 + n^{\frac{1}{c-1}}d/c^2)$ is necessary for every algorithm that computes a $c/1.01$ approximation, for every integer $c \geq 2$ (**Theorems 4.8 and 4.1**). Up to poly-logarithmic factors, our algorithm is thus optimal for every poly-logarithmic c.

In insertion-deletion streams, we give a one-pass c-approximation streaming algorithm with space $\tilde{O}(\frac{dn}{c^2})$ if $c \leq \sqrt{n}$, and space $\tilde{O}(\frac{\sqrt{n}d}{c})$ if $c > \sqrt{n}$ that succeeds w.h.p. (**Theorem 5.4**). This result yields a $O(\sqrt{n})$-approximation semi-streaming algorithm for Star Detection (**Corollary 5.5**). We complement our algorithm with a lower bound showing that space $\tilde{\Omega}(\frac{dn}{c^2})$ is required (**Theorem 6.4**), which renders our algorithm optimal (if $c \leq \sqrt{n}$) up to poly-logarithmic factors.

1.3 Further Related Work

Neighborhood Detection shares similarities with covering problems such as Set Cover since the node of largest degree covers most vertices. Almost all streaming algorithms for Set Cover [21, 22, 14, 4, 28, 3] assume that the input stream consists of the sets themselves. One exception is [31], where an *edge-arrival* setting is considered, and the input stream consists of a sequence of tuples (i, j) indicating that element i belongs to set j. Our results show that in this setting it is even impossible to find or approximate the set that covers most items in a single pass using small space, suggesting that the edge-arrival setting is strictly harder than the setting where entire sets arrive one by one.

Recently, it was shown that linear sketches are universal for the class of insertion-deletion streaming algorithms that can handle streams of length $\Omega(\ell^2)$, where ℓ is the dimension of the vector described by the input stream [30] (see also [37, 2])[4]. A consequence of these results is that lower bounds for insertion-deletion streaming algorithms can also be proved in the *simultaneous model of communication*, where multiple parties each send a message to a referee, who then outputs the result. This approach has, for example, been used to prove lower bounds for Maximum Matching in insertion-deletion streams [33, 5]. We note that two-party communication lower bounds such as the one given in this paper are more general since they also apply to algorithms that rely on short input streams.

1.4 Outline

We start with an extensive technical overview in Section 2 that summarizes all the results presented in this paper. In Section 3, we give our algorithm for insertion-only streams, and in Section 4, we present our lower bound for these streams. Our algorithm for insertion-deletion streams is given in Section 5, and we conclude with a matching lower bound in Section 6.

[3] We say that an event occurs with high probability (in short: w.h.p.) if it happens with probability at least $1 - \frac{1}{n}$, where n is a suitable parameter associated with the input size.

[4] For graphs on n vertices, the dimension can be as large as $\Theta(n^2)$ since such a graph may have up to $\Theta(n^2)$ edges. The class of relevant algorithms is thus those that can handle streams of length $\Omega(n^4)$.

2 Technical Overview

We consider simple bipartite graphs $G = (A, B, E)$ with $|A| = n$ and $|B| = m = \text{poly}(n)$. The maximum degree of an A-node is denoted by Δ. We say that a tuple $(a, S) \in A \times 2^B$ is a *neighborhood* in G if $S \subseteq \Gamma(a)$. The size $|(a, S)|$ of (a, S) is defined as $|(a, S)| = |S|$. Using this terminology, the objective of Neighborhood Detection is to output a neighborhood of size at least d/c.

Let A be a random variable distributed according to $\mathcal{D}$. The *Shannon Entropy* of A is denoted by $H_{\mathcal{D}}(A)$, or simply $H(A)$ if the distribution $\mathcal{D}$ is clear from the context. The *mutual information* of two jointly distributed random variables A, B with distribution $\mathcal{D}$ is denoted by $I_{\mathcal{D}}(A, B) := H_{\mathcal{D}}(A) - H_{\mathcal{D}}(A \mid B)$ (again, $\mathcal{D}$ may be dropped), where $H_{\mathcal{D}}(A \mid B)$ is the entropy of A conditioned on B. For an excellent overview on information theory we refer the reader to [20].

2.1 Communication Complexity

We now provide the necessary context on communication complexity (see [36] for more information).

In the *one-way p-party communication model*, for $p \geq 2$, p parties $P_1, P_2, \ldots, P_p$ communicate with each other to jointly solve a problem. Each party P_i holds their own private input X_i and has access to both private and public random coins. Communication is one-way: P_1 sends a message M_1 to P_2, who then sends a message M_2 to P_3. This process continues until P_p receives a message M_{p-1} from P_{p-1} and then outputs the result.

The way the parties interact is specified by a communication protocol Π. We say that Π is an ϵ-error protocol for a problem Prob if it is correct with probability $1-\epsilon$ on any input $(X_1, X_2, \ldots, X_p)$ that is valid for Prob, where the probability is taken over the randomness (both private and public) used by the protocol. The *communication cost* of Π is the size of the longest message sent by any of the parties, that is, $\max_{1 \leq i \leq p-1}\{|M_i|\}$, where $|M_i|$ is the maximum length of message M_i. The *randomized one-way communication complexity* $R^{\rightarrow}_{\epsilon}(\mathsf{Prob})$ of a problem Prob is the minimum communication cost among all ϵ-error protocols Π.

Let $\mathcal{D}$ be any input distribution for a specific problem Prob. The *distributional one-way communication complexity* of Prob, denoted $D^{\rightarrow}_{\mathcal{D},\epsilon}(\mathsf{Prob})$, is the minimum communication cost among all deterministic communication protocols for Prob that succeed with probability at least $1 - \epsilon$, where the probability is taken over the inputs $\mathcal{D}$. In order to prove lower bounds on $R^{\rightarrow}_{\epsilon}(\mathsf{Prob})$, by Yao's lemma it is enough to bound the distributional communication complexity for any suitable input distribution since $R^{\rightarrow}_{\epsilon}(\mathsf{Prob}) = \max_{\mathcal{D}} D^{\rightarrow}_{\mathcal{D},\epsilon}(\mathsf{Prob})$. In our lower bound arguments we will therefore consider deterministic protocols with distributional error. This is mainly for convenience as this allows us to disregard public and private coins. We note however that with additional care about private and public coins, our arguments also directly apply to randomized protocols.

Our lower bound arguments follow the *information complexity* paradigm. There are various definitions of information complexity (e.g. [7, 6, 13]), and for the sake of simplicity we will in fact omit a precise definition. Information complexity arguments typically measure the amount of information revealed by a communication protocol about the inputs of the participating parties. This quantity is a natural lower bound on the total amount of communication, as the amount of information revealed cannot exceed the number of bits exchanged. We will follow this approach in that we give lower bounds on quantities of the form $I_{\mathcal{D}}(X_i \ : \ M_j)$, for some $j \geq i + 1$. This then implies a lower bound on the communication complexity of a specific problem Prob since $I_{\mathcal{D}}(X_i \ : \ M_j) \leq H_{\mathcal{D}}(M_j) \leq |M_j|$ holds for any protocol.

2.2 Insertion-only Streams

2.2.1 One-pass Streaming Algorithm

Insertion-only streaming algorithms for Neighborhood Detection are faced with the following challenge: An A-vertex of large degree needs to be detected, and, at the same time, its incident edges need to be stored. Since, however, we will only know that an A-vertex is of large degree once we have already seen many of its incident edges, we will necessarily miss some of these. The challenge is thus to minimize the number of missed edges. We will prove the following theorem:

Theorem 3.2 (restated) *Suppose that the input graph $G = (A, B, E)$ contains at least one A-node of degree at least d. For every integral $c \geq 2$, there is a randomized one-pass streaming algorithm that finds a neighborhood of size $\frac{d}{c}$ with probability at least $1 - \frac{1}{n}$ and uses space $\mathrm{O}(n \log n + n^{\frac{1}{c}} d \log^2 n)$.*

For every $i \in \{0, 1, \ldots, c-1\}$, our algorithm runs the following strategy in parallel: First, sample uniformly at random $\Theta(n^{1/c} \log n)$ A-vertices from the set of nodes of degree at least $\frac{id}{c}$ using reservoir sampling [43] combined with degree counts. This sampling process *consumes* the first $\frac{id}{c}$ edges of every sampled vertex. Then, for every sampled vertex, we store the next d/c of its incident edges (or fewer in case there are not that many left). In more detail, our algorithm maintains the degrees of all A-vertices. As soon as the degree of an A-vertex exceeds $\frac{id}{c}$ it is inserted with a suitable probability into a reservoir of size $\Theta(n^{1/c} \log n)$ (and another vertex may then be removed from the reservoir), and we then collect up to d/c edges incident to every vertices in the reservoir.

Concerning space, $\mathrm{O}(n \log n)$ bits are needed to maintain vertex degrees. As we store at most $\frac{d}{c}$ edges for every sampled vertex, and we sample overall $\mathrm{O}(c \cdot n^{1/c} \log n)$ vertices, we obtain a total space of $\mathrm{O}(n \log n + d \cdot n^{1/c} \log^2 n)$ (accounting space $\mathrm{O}(\log n)$ for every edge).

To see why this algorithm succeeds, denote by A_i the set of A-vertices of degree at least $i \cdot d/c$ and observe that $A = A_0 \supseteq A_1 \supseteq \ldots A_{c-1}$. Furthermore, let S_i denote the set of sampled vertices for run i of our algorithm. Observe that S_i is a uniform random subset of A_i of size $\Theta(n^{\frac{1}{c}} \log n)$.

Consider first the run for $i = 0$: Then S_0 is a uniform random sample of size $\Theta(n^{1/c} \log n)$ of all A-vertices. Observe that if there were $\Omega(n^{1-\frac{1}{c}})$ A-vertices of degree at least $\frac{d}{c}$, i.e., $|A_1| = \Omega(n^{1-\frac{1}{c}})$, then this run would succeed as S_0 contained such a node w.h.p. and d/c incident edges of such a node would subsequently be stored. However, if this run fails, then we are guaranteed that there are $\mathrm{O}(n^{1-\frac{1}{c}})$ A-vertices of degree at least $\frac{d}{c}$ (i.e., $|A_1| = \mathrm{O}(n^{1-\frac{1}{c}})$). Observe that the run for $i = 1$ samples $\Theta(n^{1/c} \log n)$ vertices from exactly this set of vertices A_1. By the same argument as before, this run would succeed if $\Omega(n^{1-\frac{2}{c}})$ of these vertices had in fact a degree of at least $\frac{2d}{c}$, i.e., $|A_2| = \Omega(n^{1-\frac{2}{c}})$. Generalizing, we see that if all runs for $i \in \{0, 1, \ldots, c-2\}$ failed, then run $i = c-1$ would succeed if $|A_c| = \Omega(1)$ - in fact, parameters are chosen so that we obtain the condition $|A_c| \geq 1$. Since we are guaranteed that there is at least one vertex of degree d, the inequality $|A_c| \geq 1$ indeed holds and run $i = c-1$ would therefore succeed if all other runs failed.

2.2.2 Lower Bound for One-pass Streaming Algorithms

Concerning our lower bound for insertion-only streams, observe that if the edges of any graph are arbitrarily partitioned among k parties, then one of them necessarily holds at least a k-fraction of the edges incident to an A-node of maximum degree. Computing a k-approximation to Neighborhood

Detection is thus trivial in any k-party communication setting. Hence, in order to prove a c-approximation lower bound, the number of parties needs to be larger than c.

Our approach is as follows. We first define a p-party communication problem denoted Bit-Vector-Learning, and we prove a lower bound on its communication complexity. Then we argue that a streaming algorithm for Neighborhood Detection implies a communication protocol for Bit-Vector-Learning, which yields a suitable lower bound.

In Bit-Vector-Learning, the bits of n binary strings of different lengths are partitioned among p parties, and the last party is required to output at least a $p/1.01$-fraction of the bits of one of the n strings. Formally, the problem is defined as follows:

Problem 4 (Bit-Vector Learning(p, n, k) - restated) *Let $X_1 = [n]$ and for every $2 \leq i \leq p$, let X_i be a uniform random subset of X_{i-1} of size $n_i = n^{1-\frac{i-1}{p-1}}$. Furthermore, for every $1 \leq i \leq p$ and every $1 \leq j \leq n$, let $Y_i^j \in \{0,1\}^k$ be a uniform random bit-string if $j \in X_i$, and let $Y_i^j = \epsilon$ (the empty string) if $j \notin X_i$. For $j \in [n]$, let $Z^j = Y_1^j \circ Y_2^j \circ \cdots \circ Y_p^j$ be the bit string obtained by concatenation. Party i holds X_i and $Y_i := Y_i^1, \ldots, Y_i^n$. Communication is one-way and party p needs to output an index $I \in [n]$ and at least $1.01k$ bits from string Z^I.*

An instance of Bit-Vector Learning$(3, 4, 5)$ is illustrated in Figure 1.

Alice	$\xrightarrow{M_1}$	**Bob**	$\xrightarrow{M_2}$	**Charlie**
$X_1 = \{1,2,3,4\}$		$X_2 = \{1,4\}$		$X_3 = \{4\}$
$Y_1^1 = 10010$		$Y_2^1 = 11011$		$Y_3^1 = \epsilon$
$Y_1^2 = 01000$		$Y_2^2 = \epsilon$		$Y_3^2 = \epsilon$
$Y_1^3 = 01011$		$Y_2^3 = \epsilon$		$Y_3^3 = \epsilon$
$Y_1^4 = 01111$		$Y_2^4 = 01010$		$Y_3^4 = 00011$

Figure 1: Example instance of Bit-Vector Learning$(3, 4, 5)$. Charlie needs to output at least $1.01 \cdot 5$ positions of one of the strings $Z^1 = 1001011011$, $Z^2 = 01000$, $Z^3 = 01011$, or $Z^4 = 011110101000011$.

Observe that even without communication party p already knows a p-fraction of string Z^j, where j is the unique element in X_p. We will show that outputting slightly more than a p-fraction of one the strings requires a large amount of communication, which constitutes our main result:

Theorem 4.7. (restated) *For every $\epsilon < 0.005$, the randomized one-way communication complexity of Bit-Vector Learning(p, n, k) is bounded as follows:*

$$R_\epsilon^{\rightarrow}(\text{Bit-Vector Learning}(p, n, k)) = \Omega(\frac{kn^{\frac{1}{p-1}}}{p}) \ .$$

Bit-Vector-Learning could be solved if any party $i \in [p-1]$ managed to send at least $0.01k$ bits of Y_i^j to party $i+1$, for some $j \in X_{i+1}$. However, since party i has no knowledge of the set X_{i+1}, on an intuitive level they can therefore only *guess* which of their bits are relevant. The sizes of the sets $X_1, \ldots, X_p$ are however chosen so that many guesses are required, yielding large messages.

Our key lemma reflecting this intuition is as follows: (we use the notation $Y_{i-1}^{X_i}$ to mean the strings $Y_{i-1}^{x_1}, Y_{i-1}^{x_2}, \ldots$ where $x_1, x_2, \ldots$ are the elements of X_i)

Lemma 4.5. (restated) *The following inequality holds:*

$$I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i) \leq \frac{|M_{i-1}|}{n^{\frac{1}{p-1}}} \ .$$

The previous lemma shows that M_{i-1} carries only little information about the relevant strings $Y_{i-1}^{X_i}$. If we wanted party i to learn at least $0.01k$ bits from any of the bit-strings Y_{i-1}^{j}, for some $j \in X_i$, then a message of size at least $n^{\frac{1}{p-1}} \cdot 0.01k = \Omega(n^{\frac{1}{p-1}}k)$ would be needed. Since there are p parties, there are $p-1$ possibilities for the parties to learn $0.01k$ relevant bits. This explains on an intuitive level why our final lower bound given in Theorem 4.7 is $\Omega(n^{\frac{1}{p-1}}k/p)$.

The proof of Lemma 4.5 relies on a combination of the chain rule for mutual information and Baranyai's theorem concerning the coloring of complete regular hypergraphs [8], which can also be stated as follows: Let S be the set of all k-subsets of $[n]$, for some k that divides n. Then S can be partitioned into $|S|\frac{k}{n} = \binom{n}{k}\frac{k}{n}$ sets $S_1, S_2, \ldots$ such that $|S_i| = \frac{n}{k}$ and $\cup_{x \in S_i} x = [n]$ holds for every i. First, using elementary information theoretic arguments, we rewrite (for justifications of each individual inequality see the proof in Section 4.4):

$$\begin{aligned} I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i) &\leq I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i X_{i-1}) \\ &= \mathop{\mathbb{E}}_{x_{i-1} \leftarrow X_{i-1}} \mathop{\mathbb{E}}_{x_i \leftarrow X_i} I(M_{i-1} \ : \ Y_{i-1}^{x_i} \mid X_{i-1} = x_{i-1}) \ . \end{aligned} \tag{1}$$

Next, for any fixed value x_{i-1} for X_{i-1}, the random variable X_i is a uniform random subset of x_{i-1} of size n_i. Let $\mathcal{X}(x_{i-1})$ be the set of all n_i-subsets of x_{i-1}. Then, by Baranyai's theorem, we can partition $\mathcal{X}(x_{i-1})$ into $|\mathcal{X}(x_{i-1})|\frac{n_i}{n_{i-1}}$ subsets $\mathcal{X}_1(x_{i-1}), \mathcal{X}_2(x_{i-1}), \ldots$ each of size $\frac{n_{i-1}}{n_i}$ so that $\cup_{x \in \mathcal{X}_j(x_{i-1})} x = x_{i-1}$, for every j. With some more rewriting, we obtain:

$$\begin{aligned} &\mathop{\mathbb{E}}_{x_i \leftarrow X_i} I(M_{i-1} \ : \ Y_{i-1}^{x_i} \mid X_{i-1} = x_{i-1}) \\ &\quad = \frac{1}{|\mathcal{X}(x_{i-1})|} \sum_{x_i \in \mathcal{X}(x_{i-1})} I(M_{i-1} \ : \ Y_{i-1}^{x_i} \mid X_{i-1} = x_{i-1}) \\ &\quad = \frac{1}{|\mathcal{X}(x_{i-1})|} \sum_{j \in [|\mathcal{X}(x_{i-1})|\frac{n_i}{n_{i-1}}]} \ \sum_{x_i \in \mathcal{X}_j(x_{i-1})} I(M_{i-1} \ : \ Y_{i-1}^{x_i} \mid X_{i-1} = x_{i-1}) \ . \end{aligned} \tag{2}$$

Since the elements in $\mathcal{X}_j(x_{i-1})$ precisely make up x_{i-1}, we will use the chain rule for mutual information. Denoting the elements of $\mathcal{X}_j(x_{i-1})$ by $x^1, x^2, \ldots$, we obtain

$$\begin{aligned} \sum_{x^\ell \in \mathcal{X}_j(x_{i-1})} I(M_{i-1} \ : \ Y_{i-1}^{x^\ell} \mid X_{i-1} = x_{i-1}) &\leq \sum_{x^\ell \in \mathcal{X}_j(x_{i-1})} I(M_{i-1} \ : \ Y_{i-1}^{x^\ell} \mid x^1, \ldots, x^{\ell-1}, \ X_{i-1} = x_{i-1}) \\ &= I(M_{i-1} \ : \ Y_{i-1} \mid X_{i-1} = x_{i-1}) \ . \end{aligned} \tag{3}$$

Then, from Inequalities 3, 2, and 1 we obtain

$$I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i) \ \leq \ \frac{|\mathcal{X}(x_{i-1})|\frac{n_i}{n_{i-1}}}{|\mathcal{X}(x_{i-1})|} \cdot I(M_{i-1} \ : \ Y_{i-1} \mid X_{i-1}) \leq \frac{n_i}{n_{i-1}} H(M_{i-1}) \leq \frac{n_i}{n_{i-1}}|M_{i-1}| \ ,$$

which completes the informal proof of Lemma 4.5.

Last, it remains to argue that a streaming algorithm for Neighborhood Detection can be used for solving Bit-Vector-Learning. Indeed, based on their inputs for Bit-Vector-Learning(p, n, k), the p parties construct a bipartite graph $G = (A, \dot{\cup}_{i\in[p]} B_i, \dot{\cup}_{i\in[p]} E_i)$, where $A = [n]$ and $|B_i| = 2k$, for every i. Party i adds edges E_i between the A-vertices and the set B_i. This is done as follows: For each $j \in X_i$, vertex a_j is connected to k of the $2k$ vertices in B_i. The $2k$ vertices in B_i are regarded as k 2-tuples of vertices, where each tuple corresponds to one position in the bit strings Y_i. Then, depending on the value of the bit $Y_i^j[\ell]$, for every $1 \leq \ell \leq k$, a_j is either connected to the first or the second vertex of the ℓth tuple. Observe that $\Delta = kp$ in this construction.

See Figure 2 for an illustration of the example given in Figure 1.

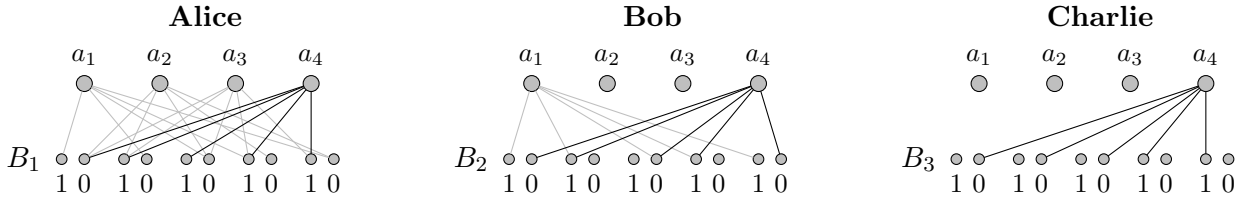

Figure 2: In the example instance given in Figure 1, Alice holds $Y_1^1 = 10010$, $Y_1^2 = 01000$, $Y_1^3 = 01011$, and $Y_1^4 = 01111$. For each string Y_1^j, Alice connects vertex a_j to 5 vertices, each indicating one bit of the respective bit string. For example, when reading the labels of the B-vertices connected to a_4 from left-to-right, we obtain the bit sequence 01111 which equals Y_1^4.

The parties then simulate any $\frac{p}{1.01}$-approximation streaming algorithm $\mathbf{A}$ for Neighborhood Detection with parameter $d = \Delta = kp$ as follows: The first party runs $\mathbf{A}$ on edges E_1 and sends the resulting memory state to the second party, who then continues the algorithm on E_2. This process continues until party p completes the algorithm and thus computes a neighborhood of size at least $1.01d/p = 1.01k$. Since each edge of this neighborhood allows us to identify one position in the respective bit string, we obtain a solution to Bit-Vector-Learning. This yields our lower bound result:

Theorem 4.8. (restated) *Let* $\mathbf{A}$ *be a c-approximation streaming algorithm for Neighborhood Detection with error probability at most* 0.005 *with* $c = \frac{p}{1.01}$*, for some integer* $p \geq 2$*. Then* $\mathbf{A}$ *uses space at least:*

$$\Omega(\frac{dn^{\frac{1}{p-1}}}{c^2}) \ .$$

2.3 Insertion-deletion Streams

2.3.1 One-pass Streaming Algorithm

The approach taken for insertion-only streams cannot be applied to insertion-deletion streams since vertex degrees may decrease over the course of the algorithm. Essentially all known insertion-deletion streaming algorithms are solely based on the computation of linear sketches, and our algorithm is no exception. An l_0-*sampler* in insertion-deletion streams outputs a uniform random element from the non-zero coordinates of the vector described by the input stream, and implemen-

tations of l_0-sampling are known that require poly-logarithmic space [32]. In our setting, the input vector is of dimension $n \cdot m$ where each coordinate indicates the presence or absence of an edge.

l_0-sampling allows us, for example, to sample uniformly at random from the edges of the input graph, or, by considering the substream of edges incident to a specific vertex, to sample from the edges incident to a specific vertex. Our algorithm combines these two strategies:

1. **Vertex Sampling.** Sample u.a.r. $\Theta(\frac{n}{c}\log n)$ A-vertices before processing the stream. Then, for each sampled vertex, sample $\Theta(\frac{d}{c}\log n)$ edges (with repetition) from its incident edges using l_0-sampling. This yields at least $\frac{d}{c}$ different edges if the degree of the sampled node is at least $\frac{d}{c}$. This strategy uses space $\tilde{\mathrm{O}}(\frac{nd}{c^2})$ and yields a c-approx. to Neighborhood Detection if the input graph contains $\Omega(c)$ nodes of degree at least $\frac{d}{c}$, since then one of these nodes would be sampled.

2. **Edge Sampling.** Sample $\tilde{\Theta}(\frac{nd}{c^2})$ edges from the input stream using l_0-sampling. Observe that if the vertex sampling strategy does not succeed, then we are guaranteed that the input graph contains $\mathrm{O}(c)$ A-vertices of degree at least $\frac{d}{c}$. This implies that the input graph has $\mathrm{O}\left(c \cdot \Delta + n \cdot \frac{d}{c}\right)$ edges. The probability that an l_0-sampler returns an edge incident to a distinguished node of degree Δ is hence $\Omega(\frac{\Delta}{c\cdot\Delta + n\cdot\frac{d}{c}})$, and since we run $\tilde{\Theta}(\frac{nd}{c^2})$ l_0-samplers, we expect

$$\Omega\left(\frac{\Delta}{c\cdot\Delta + n\cdot\frac{d}{c}}\cdot\frac{nd}{c^2}\right) = \Omega\left(\frac{\Delta n}{c^2\Delta + dn}\cdot\frac{d}{c}\right) = \Omega\left(\frac{d}{c}\right) \ ,$$

   l_0-samplers to return edges incident to a node of degree Δ, using the fact that $c^2 \leq n$ and $\Delta \geq d$.

Conducting a more careful analysis, we obtain the following theorem:

Theorem 5.4. (restated) *There is a one-pass c-approximation streaming for insertion-deletion streams that uses space $\tilde{\mathrm{O}}(\frac{dn}{c^2})$ if $c \leq \sqrt{n}$, and space $\tilde{\mathrm{O}}(\frac{\sqrt{n}d}{c})$ if $c > \sqrt{n}$ and succeeds w.h.p.*

2.3.2 Lower Bound for One-pass Streaming Algorithms

Our lower bound for insertion-deletion streaming algorithms relies on proving a lower bound for a new two-party communication problem denoted Augmented-Matrix-Row-Index, which can be seen as an extension of the well-known Augmented Index problem to two dimensions. We use the following notation: Let X be an n-by-m matrix. Then the ith row of X is denoted X_i. A position (i, j) is a tuple chosen from $[n] \times [m]$. We will index the matrix X by a set of positions S, i.e., X_S, meaning the matrix positions $X_{i,j}$, for every $(i, j) \in S$.

Problem 5. (Augmented-Matrix-Row-Index(n, m, k) - restated) *In Augmented-Matrix-Row-Index, Alice holds a binary matrix $X \in \{0, 1\}^{n\times m}$ where every X_{ij} is a uniform random Bernoulli variable, for some integers n, m. Bob holds a uniform random index $J \in [n]$ and for each $i \neq J$, Bob holds a uniform random subset of positions $Y_i \subseteq \{i\} \times [m]$ with $|Y_i| = m - k$ and also knows X_{Y_i}. Alice sends a message to Bob who then outputs the entire row X_J.*

For ease of notation, we define $Y_I = \perp$ and $Y = Y_1, Y_2, \ldots, Y_n$. An example instance of Augmented-Matrix-Row-Index$(4, 6, 2)$ is given in Figure 3. As our main theorem, we prove a lower bound on the one-way communication complexity of Augmented-Matrix-Row-Index (ϵ denotes the error probability of the protocol):

Theorem 6.2. (restated) *Let $\epsilon \leq \frac{k}{2m}$. Then:*

$$R^{\rightarrow}_{\epsilon}(\textsf{Augmented-Matrix-Row-Index}(n,m,k)) = \Omega(nk) \ .$$

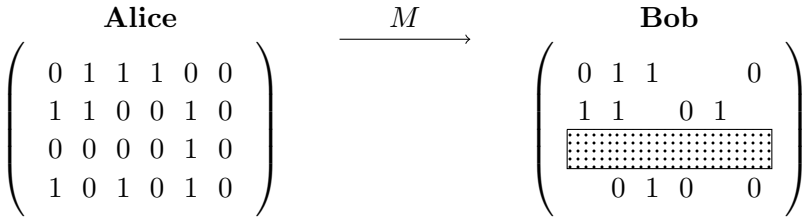

Figure 3: Example Instance of $\textsf{Augmented-Matrix-Row-Index}(4,6,2)$. Bob needs to output the content of row 3. Bob knows $6-2=4$ random positions in every row except row 3.

Our goal is to bound the term $I(X \ : \ M)$ from below, which then also bounds $|M|$. To this end, first, using a Fano-type argument, since Bob's knowledge, i.e., the variables M, J, Y, X_Y, allows Bob to determine row X_I, we show that $I(X_J \ : \ MJYX_Y) = \Omega(m)$, which further implies that

$$I(X_{\tilde{Z}_J} \ : \ M \mid JYX_Y) = \Omega(k) \ , \tag{4}$$

where $\tilde{Z}_J$ are arbitrary k positions in row J. This inequality will be used later.

Next, for $i \neq J$ let Z_i denote all positions of row i unknown to Bob, i.e., $Z_i = (\{i\} \times [m]) \setminus Y_i$, let $Z_J = \perp$, and let $Z = Z_1, Z_2, \ldots, Z_n$. We partition the matrix X as follows:

$$\begin{aligned} I(X \ : \ M) &= I(X_Y X_J X_Z \ : \ M) = I(X_Y X_J \ : \ M) + I(X_Z \ : \ M \mid X_J X_Y) \\ &\geq I(X_Z \ : \ M \mid X_J X_Y) \ . \end{aligned}$$

Our goal is to show that the information about the parts unknown to Bob in each row different to the row J is large. To this end, let L be a variable that is uniformly distributed in $[n] \setminus I$. Then:

$$\begin{aligned} I(X_Z \ : \ M \mid X_J X_Y) &= \sum_{i \in [n] \setminus J} I(X_{Z_i} \ : \ M \mid \ X_J X_Y X_{Z_1}, \ldots, X_{Z_{i-1}}) \\ &\geq \sum_{i \in [n] \setminus J} I(X_{Z_i} \ : \ M \mid \ X_J X_Y) = (n-1) \cdot I(X_{Z_L} \ : \ M \mid \ X_J X_Y L) \ . \end{aligned}$$

It remains to show that $I(X_{Z_L} \ : \ M \mid \ X_J X_Y L) = \Omega(k)$. This expression measures the amount of information about k random positions in a random row L contained in M conditioned on knowing various bits in all rows. Observe that this expression is similar to the one in Equality 4 with a slightly different conditioning. In our full argument, we exploit this similarity to complete the proof.

Last, we show that a c-approximation algorithm for $\textsf{Neighborhood Detection}(n,d)$ with space s yields a communication protocol for $\textsf{Augmented-Matrix-Row-Index}(n, 2d, \frac{d}{c} - 1)$ with message size $\mathrm{O}(s \cdot c \cdot \log n)$. To this end, Alice and Bob permute each row i of matrix X independently using random permutations π_i chosen from public randomness. Assuming that row J contains at least d 1s, we show that a c-approximation algorithm for $\textsf{Neighborhood Detection}(d,c)$ allows Bob to learn a uniform random $\Theta(c)$-fraction of the 1s in row J. By concentration bounds, repeating this

algorithm $\Theta(c \log n)$ times, Bob learns all 1s in row J, and therefore knows that the remaining entries are 0s. To cover the case when row J contains fewer than d 1s, we run the same protocol on the matrix where every entry is inverted. This yields the following result:

Theorem 6.4. (restated) *Every c-approximation insertion-deletion streaming algorithm for* Neighborhood Detection(n, d) *that fails with probability* $\delta \leq \frac{1}{2d}$ *requires space* $\Omega\left(\frac{nd}{c^2 \log n})\right)$.

3 Algorithm for Insertion-only Streams

Before presenting our algorithm for Neighborhood Detection in insertion-only streams, we discuss a sampling subroutine that combines reservoir sampling with degree counts.

3.1 Degree-based Reservoir Sampling

The subroutine Deg-Res-Sampling(d_1, d_2, s) samples s nodes uniformly at random from the set of nodes of degree at least d_1, and for each of these nodes computes a neighborhood of size $\min\{d_2, \deg - d_1 + 1\}$, where deg is the degree of the respective node. If at least one neighborhood of size d_2 is found then we say that the algorithm *succeeds* and returns a uniform random neighborhood among the stored neighborhoods of sizes d_2. Otherwise, we say that the algorithm *fails* and it reports `fail`.

This is achieved as follows: While processing the stream of edges, the degrees of all A-vertices are maintained. The algorithm maintains a reservoir of size s that fulfills the invariant that at any moment it contains a uniform sample of size s of the set of nodes whose current degrees are at least d_1 (or, in case there are fewer than s such nodes, it contains all such nodes). To this end, as soon as the degree of an A-vertex reaches d_1, the vertex is introduced into the reservoir with an appropriate probability (and another vertex is removed if the reservoir is already full), so as to maintain a uniform sample. Once a vertex is introduced into the reservoir, incident edges to this vertex are collected until at most d_2 such edges are found.

The description of Algorithm 1 assumes that we have a function Coin(p) to our disposal that outputs `true` with probability p and `false` with probability $1 - p$.

Disregarding the maintenance of the vertex degrees, the algorithm uses space $O(sd_2 \log n)$ since at most d_2 neighbors for each vertex in the reservoir are stored, and we account space $\mathrm{O}(\log n)$ for storing an edge.

Lemma 3.1. *Suppose that G contains at most n_1 A-nodes of degree at least d_1 and at least n_2 A-nodes of degree at least $d_1 + d_2 - 1$. Then, Algorithm* Deg-Res-Sampling(d_1, d_2, s) *succeeds with probability at least*

$$1 - (1 - \frac{s}{n_1})^{n_2} \geq 1 - e^{-\frac{sn_2}{n_1}} \ .$$

Proof. Let $D \subseteq V$ be the set of vertices of degree at least d_1 (then $|D| \leq d_1$). First, suppose that $d_1 \leq s$. Then the algorithm stores all nodes of degree at least d_1 (including all nodes of degree $d_1 + d_2 - 1$) and collects its incident edges (except the first $d_1 - 1$ such edges). It therefore necessarily finds a neighborhood of size d_2.

Otherwise, by well-known properties of reservoir sampling (e.g. [43]), at the end of the algorithm the set R constitutes a uniform random sample of D of size s. The probability that no node of

Algorithm 1 DEG-RES-SAMPLING(d_1, d_2, s)

Require: Integral degree bounds d_1 and d_2, reservoir size s

$R \leftarrow \{\}$ {reservoir}, $S \leftarrow \{\}$ {collected edges}, $x \leftarrow 0$ {counter for nodes of degree $\geq d_1$}
while stream not empty **do**
  Let ab be next edge in stream
  Increment degree $\deg(a)$ by one
  if $\deg(a) = d_1$ **then** {candidate to be inserted into reservoir}
    $x \leftarrow x + 1$
    if $|R| < s$ **then** {Reservoir not yet full}
      $R \leftarrow R \cup \{a\}$
    else {Reservoir full}
      if COIN$(\frac{s}{x})$ **then** {Insert a into reservoir with probability $\frac{s}{x}$}
        Let a' be a uniform random element in R
        $R \leftarrow (R \setminus \{a'\}) \cup \{a\}$, delete all edges incident to a' from S
  if $a \in R$ and $\deg_S(a) < d_2$ **then** {collect edges incident to vertices in R}
    $S \leftarrow S \cup \{ab\}$
return Random neighborhood among those of size d_2 in S, if none exists return `fail`

degree at least $d_1 + d_2 - 1$ is sampled is at most:

$$\begin{aligned}\frac{\binom{n_1-n_2}{s}}{\binom{n_1}{s}} &= \frac{(n_1-n_2)!(n_1-s)!}{(n_1-n_2-s)!n_1!} = \frac{(n_1-s)\cdot(n_1-s-1)\cdot\ldots\cdot(n_1-s-n_2+1)}{n_1\cdot(n_1-1)\cdot\ldots\cdot(n_1-n_2+1)} \\ &\leq \left(\frac{n_1-n_2-s+1}{n_1-n_2+1}\right)^{n_2} = (1-\frac{s}{n_1-n_2+1})^{n_2} \leq e^{-\frac{sn_2}{n_1-n_2+1}} \leq e^{-\frac{sn_2}{n_1}} ,\end{aligned}$$

where the last inequality holds for $n_2 \geq 1$ (which is always the case).

□

3.2 Main Algorithm

Our main algorithm runs the subroutine presented in the previous subsection in parallel for multiple different threshold values d_1. We will prove that the existence of a node of degree d implies that at least one of these runs will succeed with high probability.

Algorithm 2 One-pass c-approximation Streaming Algorithm for Neighborhood Detection

Require: Space s, degree bound d

$s \leftarrow \lceil \ln(n) \cdot n^{\frac{1}{c}} \rceil$
for $i = 0 \ldots c-1$ **do in parallel**
  $(a_i, S_i) \leftarrow$ DEG-RES-SAMPLING$(\max\{1, i \cdot \frac{d}{c}\}, \frac{d}{c}, s)$
return Uniform random neighborhood (a_i, S_i) among those runs that succeeded

Theorem 3.2. *Suppose that the input graph $G = (A, B, E)$ contains at least one A-node of degree at least d. For every integral $c \geq 2$, Algorithm 2 finds a neighborhood of size $\frac{d}{c}$ with probability at least $1 - \frac{1}{n}$ and uses space*

$$\mathrm{O}(n \log n + n^{\frac{1}{c}} d \log^2 n) .$$

Proof. Concerning the space bound, the algorithm needs to keep track of the degrees of all A-vertices which requires space $O(n \log n)$ (using the assumption $m = \text{poly}\, n$). The algorithm runs the subroutine Algorithm 1 c times in parallel. Each of these runs requires space $\mathrm{O}(s \cdot \frac{d}{c} \log n)$. Besides the vertex degrees, we thus have an additional space requirement of $O(s \cdot d \log n) = O(n^{\frac{1}{c}} d \log^2 n)$ bits, which justifies the space requirements.

Concerning correctness, let n_0 be the number of A-nodes of degree at least 1, and for $i \geq 1$, let n_i be the number of A-nodes of degree at least $i \cdot \frac{d}{c}$. Observe that $n \geq n_0 \geq n_1 \geq n_2 \geq \cdots \geq n_c \geq 1$, where the last inequality follows from the assumption that the input graph contains at least one A-node of degree at least d.

We will prove that at least one of the runs succeeds with probability at least $1 - \frac{1}{n}$. For the sake of a contradiction, assume that the error probability of every run is strictly larger than $\frac{1}{n}$. Then, using Lemma 3.1, we obtain for every $0 \leq i \leq c-1$:

$$\begin{aligned} e^{-\frac{sn_{i+1}}{n_i}} &> \frac{1}{n}\ , \text{ which implies} \\ n_{i+1} &< \frac{\ln(n) n_i}{s}\ . \end{aligned}$$

Since $n_0 \leq n$ we obtain:

$$n_i < n \left(\frac{\ln n}{s}\right)^i\ ,$$

and since $n_c \geq 1$ we have:

$$1 < n \left(\frac{\ln n}{s}\right)^c \quad \text{which implies} \quad s < n^{\frac{1}{c}} \ln n\ .$$

However, since the reservoir size in Algorithm 2 is chosen to be $\lceil n^{\frac{1}{c}} \ln n \rceil$, we obtain a contradiction. Hence, at least one run succeeds with probability $1 - 1/n$. □

The previous result can be used to obtain a semi-streaming algorithm for Star Detection.

Corollary 3.3. *There is a semi-streaming* $\mathrm{O}(\log n)$*-approximation algorithm for Star Detection that succeeds with high probability.*

Proof. Let $G = (V, E)$ be the graph described by the input stream in an instance of Star Detection. We use $\mathrm{O}(\log_{1+\epsilon} n)$ guesses $\Delta' \in \{1, 1+\epsilon, (1+\epsilon)^2, \ldots, (1+\epsilon)^{\lceil \log_{1+\epsilon} n \rceil}\}$ for Δ, the maximum degree in the input graph. For each guess Δ' we run our insertion-only algorithm for Neighborhood Detection with threshold value $d = \Delta'$ on the bipartite graph $H = (V, V, E')$, where for every edge uv in the input stream, we include the two edges uv and vu into H.

Consider the run with the largest value for Δ' that is not larger than Δ. Then, $\Delta' \geq \Delta/(1+\epsilon)$. This run detects a neighborhood of size at least $\Delta'/c \geq \Delta/(c(1+\epsilon))$ and thus a star of this size in G. We thus obtain a $(1+\epsilon)c$-approximation algorithm with space $\tilde{\mathrm{O}}(\log_{1+\epsilon}(n) n^{1+\frac{1}{c}})$. Using any fixed constant for ϵ and $c = \log n$, this construction yields a $(1+\epsilon)\log n$-approximation semi-streaming algorithm for approximating the largest star in a general graph. □

4 Lower Bound for Insertion-only Streams

In this section, we first point out that a simple $\Omega(n/c^2)$ lower bound follows from the one-way communication complexity of a multi-party version of the Set-Disjointness problem. Next, we give some important inequalities involving entropy and mutual information that are used subsequently. Then, we prove our main lower bound result of this section. To this end, we first define the multi-party one-way communication problem Bit-Vector Learning and prove a lower bound on its communication complexity. We then show that a streaming algorithm for Neighborhood Detection yields a protocol for Bit-Vector Learning, which gives the desired lower bound.

4.1 An $\Omega(n/c^2)$ Lower Bound via Multi-party Set-Disjointness

Consider the one-way multi-party version of the well-known Set-Disjointness problem:

Problem 3 (Set-Disjointness$_p$)**.** *Set-Disjointness$_p$ is a p-party communication problem where every party i holds a subset $S_i \subseteq \mathcal{U}$ of a universe $\mathcal{U}$ of size n. The parties are given the promise that either their sets are pairwise disjoint, i.e., $S_i \cap S_j = \varnothing$ for every $i \neq j$, or they* uniquely *intersect, i.e., $|\cap_i S_i| = 1$. The goal is to determine which is the case.*

It is known that every ϵ-error protocol for Set-Disjointness$_p$ requires a total communication of $\Omega(n/p)$ bits [12]. Since our notion of one-way multi-party communication complexity measures the maximum length of any message sent in an optimal protocol, we obtain:

$$R^{\rightarrow}_{\epsilon}(\textsf{Set-Disjointness}_p) = \Omega(n/p^2) \ .$$

We now argue that an algorithm for Neighborhood Detection can be used to solve Set-Disjointness$_p$.

Theorem 4.1. *Every $c/1.01$-approximation streaming algorithm for Neighborhood Detection(n, d) requires space $\Omega(n/c^2)$, for any integral c and for any $d = k \cdot c$, where k is a positive integer.*

Proof. Let $S_1, \ldots, S_p$ be the sets in an instance of Set-Disjointness$_p$. For $c = p/1.01$, let $\mathbf{A}$ be a c-approximation streaming algorithm for Neighborhood Detection, and let $d = k \cdot p$, for some integer $k \geq 1$. The parties use $\mathbf{A}$ to solve Set-Disjointness$_p$ as follows: The p-parties construct a graph $G = (\mathcal{U}, B, E)$ with $B = [d]$ and $E = \cup_{i=1}^{p} E_i$. Each party i translates S_i into the set of edges E_i where for each $u \in S_i$ the edges $\{ub \ : \ b \in \{(i-1)d/p + 1, \ldots, id/p\}\}$ are included in E_i. Observe that $\Delta = d/p = k$ if all sets S_i are pairwise disjoint, and $\Delta = d = k \cdot p$ if they uniquely intersect. Party 1 now simulates $\mathbf{A}$ on their edges E_1, sends the resulting memory state to party 2 who continues running $\mathbf{A}$ on E_2. This continues until party p completes the algorithm. Since $\mathbf{A}$ is a $p/1.01$-approximation algorithm, if the sets uniquely intersect, the output of the algorithm is a neighborhood of size at least $\lceil \frac{\Delta}{c} \rceil = \lceil 1.01 \cdot k \rceil \geq k + 1$. If the sets are disjoint, then no neighborhood is of size larger than k. The last party can thus distinguish between the two cases and solve Set-Disjointness$_p$. Since at least one message used in the protocol is of length $\Omega(n/p^2)$, $\mathbf{A}$ uses space $\Omega(n/p^2) = \Omega(n/c^2)$.

□

4.2 Inequalities Involving Entropy and Mutual Information

In the following, we will use various properties of entropy and mutual information. The most important ones are listed below (let A, B, C be jointly distributed random variables):

1. *Chain Rule for Entropy:* $H(AB \mid C) = H(A \mid C) + H(B \mid AC)$
2. *Conditioning reduces Entropy:* $H(A) \geq H(A \mid B) \geq H(A \mid BC)$
3. *Chain Rule for Mutual Information:* $I(A \ : \ BC) = I(A \ : \ B) + I(A \ : \ C \mid B)$
4. *Data Processing Inequality:*[5] Suppose that C is a deterministic function of B. Then: $I(A \ : \ B) \geq I(A \ : \ C)$
5. *Independent Events:* Let E be an event independent of A, B, C. Then: $I(A \ : \ B \mid C, E) = I(A \ : \ B \mid C)$

We will also use the following claim: (see Claim 2.3. in [4] for a proof)

Lemma 4.2. *Let A, B, C, D be jointly distributed random variables so that A and D are independent conditioned on C. Then: $I(A \ : \ B \mid CD) \geq I(A \ : \ B \mid C)$.*

4.3 Hard Communication Problem: Bit-Vector Learning

We consider the following one-way p-party communication game:

Problem 4 (Bit-Vector Learning(p, n, k))**.** *Let $X_1 = [n]$ and for every $2 \leq i \leq p$, let X_i be a uniform random subset of X_{i-1} of size $n_i = n^{1-\frac{i-1}{p-1}}$. Furthermore, for every $1 \leq i \leq p$ and every $1 \leq j \leq n$, let $Y_i^j \in \{0,1\}^k$ be a uniform random bit-string if $j \in X_i$, and let $Y_i^j = \epsilon$ (the empty string) if $j \notin X_i$. For $j \in [n]$, let $Z^j = Y_1^j \circ Y_2^j \circ \cdots \circ Y_p^j$ be the bit string obtained by concatenation.*

Party i holds X_i and $Y_i := Y_i^1, \ldots, Y_i^n$. Communication is one way from party 1 through party p and party p needs to output an index $I \in [n]$ and at least $1.01k$ bits from string Z^I. [6]

Observe that the previous definition also defines an input distribution. All subsequent entropy and mutual information terms refer to this distribution. An example instance of Bit-Vector Learning$(3, 4, 5)$ is given in Figure 1 in Section 2.2.2.

In the following, for a subset $S \subseteq [n]$, we will use the notation Y_i^S, which refers to the strings $Y_i^{s_1}, Y_i^{s_2}, \ldots, Y_i^{s_{|S|}}$, where $S = \{s_1, s_2, \ldots, s_{|S|}\}$.

Observe further that there is a protocol that requires no communication and outputs an index I and k bits of Z^I: Party p simply outputs the single element $I \in X_p$ together with the bit string Y_p^I. As our main result of this section we show that every protocol that outputs at least $1.01k$ bits of any string Z^i ($i \in [n]$) needs to send at least one message of length $\Omega(\frac{kn^{\frac{1}{p-1}}}{p})$.

Remark: For technical reasons we will only consider values for n so that $n^{\frac{1}{p-1}}$ is integral. This condition implies that $n_{i+1} \mid n_i$ for every $1 \leq i \leq p-1$ since $\frac{n_i}{n_{i+1}} = n^{\frac{1}{p-1}}$. The reason for this restriction is that we will apply Baranyai's theorem [8], which is stated as Theorem 4.4 below, and requires this property.

[5] Technically the data processing inequality is more general, however, the inequality stated here is sufficient for our purposes.

[6] More formally, the output is an index $I \in [n]$ and a set of tuples $\{(i_1, \tilde{Z}_1), (i_2, \tilde{Z}_2), \ldots\}$ of size at least $1.01k$ with $i_j \neq i_k$ for every $j \neq k$ so that $Z^I[i_j] = \tilde{Z}_j$, for every j.

4.4 Lower Bound Proof for Bit-Vector Learning

Fix now an arbitrary deterministic protocol Π for Bit-Vector Learning(p, n, k) with distributional error ϵ. Let $Out = (I, \tilde{Z}^I)$ denote the neighborhood outputted by the protocol. Furthermore, denote by M_i the message sent from party i to party $i+1$. Throughout this section let $s = \max_i |M_i|$.

Since the last party correctly identifies $1.01k$ bits of Z^I, the mutual information between Z^I and all random variables known to the last party, that is, M_{p-1}, X_p and Y_p, needs to be large. This is proved in the next lemma:

Lemma 4.3. *We have:*

$$I(M_{p-1}X_pY_p \ : \ Z^I) \geq (1-\epsilon)1.01k - 1 \ .$$

Proof. We will first bound the term $I(Out \ : \ Z^I) = H(Z^I) - H(Z^I \mid Out)$. To this end, let E be the indicator variable of the event that the protocol errs. Then, $\mathbb{P}[E=1] \leq \epsilon$. We have:

$$H(E, Z^I \mid Out) \ = \ H(Z^I \mid Out) + H(E \mid Out, Z^I) = H(Z^I \mid Out) \ , \tag{5}$$

where we used the chain rule for entropy and the fact that $H(E \mid Out, Z^I) = 0$ since E is fully determined by Out and Z^I. Furthermore,

$$H(E, Z^I \mid Out) \ = \ H(E \mid Out) + H(Z^I \mid E, Out) \leq 1 + H(Z^I \mid E, Out) \ , \tag{6}$$

using the chain rule for entropy and the bound $H(E \mid Out) \leq H(E) \leq 1$ (conditioning reduces entropy). From Inequalities 5 and 6 we obtain:

$$H(Z^I \mid Out) \ \leq \ 1 + H(Z^I \mid E, Out) \ . \tag{7}$$

Next, we bound the term $H(Z^I \mid E, Out)$ as follows:

$$H(Z^I \mid E, Out) \ = \ \mathbb{P}\left[E=0\right] H(Z^I \mid Out, E=0) + \mathbb{P}\left[E=1\right] H(Z^I \mid Out, E=1) \ . \tag{8}$$

Concerning the term $H(Z^I \mid Out, E=0)$, since no error occurs, Out already determines at least $1.01k$ bits of Z^I. We thus have that $H(Z^I \mid Out, E=0) \leq H(Z^I) - 1.01k$. We bound the term $H(Z^I \mid Out, E=1)$ by $H(Z^I \mid Out, E=1) \leq H(Z^I)$ (since conditioning can only decrease entropy). The quantity $H(Z^I \mid E, Out)$ can thus be bounded as follows:

$$H(Z^I \mid E, Out) \ \leq \ (1-\epsilon)(H(Z^I) - 1.01k) + \epsilon H(Z^I) = H(Z^I) - (1-\epsilon)1.01k \ . \tag{9}$$

Next, using Inequalities 7 and 9, we thus obtain:

$$\begin{aligned} I(Out \ : \ Z^I) \ &= \ H(Z^I) - H(Z^I \mid Out) \geq H(Z^I) - 1 - H(Z^I \mid E, Out) \\ &\geq \ H(Z^I) - 1 - (H(Z^I) - (1-\epsilon)1.01k) = (1-\epsilon)1.01k - 1 \ . \end{aligned}$$

Last, observe that Out is a function of M_{p-1}, X_p, and Y_p. The result then follows from the data processing inequality. □

Next, since the set X_i is a uniform random subset of X_{i-1}, we will argue in Lemma 4.5 that the message M_{i-1} can only contain a limited amount of information about the bits $Y_{i-1}^{X_i}$. This will be stated as a suitable conditional mutual information expression that will be used later. The proof of Lemma 4.5 relies on Baranyai's theorem [8], which in its original form states that every complete regular hypergraph is 1-factorizable, i.e., the set of hyperedges can be partitioned into 1-factors. We restate this theorem as Theorem 4.4 in a form that is more suitable for our purposes.

Theorem 4.4 (Baranyai's theorem [8] - rephrased)**.** *Let k, n be integers so that $k \mid n$. Let $S \subseteq 2^{[n]}$ be the set consisting of all subsets of $[n]$ of cardinality k. Then there exists a partition of S into $|S|\frac{k}{n}$ subsets $S_1, S_2, \ldots, S_{|S|\frac{k}{n}}$ such that:*

1. *$|S_i| = \frac{n}{k}$, for every i,*
2. *$S_i \cap S_j = \varnothing$, for every $i \neq j$, and*
3. *$\bigcup_{x \in S_i} x = [n]$, for every i.*

Lemma 4.5. *Suppose that $n_i \mid n_{i+1}$. Then the following inequality holds:*

$$I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i) \leq \frac{n_i}{n_{i-1}} |M_{i-1}| \ .$$

Proof. First, using Lemma 4.2, we obtain $I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i) \leq I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i X_{i-1})$ (observe that $Y_{i-1}^{X_i}$ and X_{i-1} are independent conditioned on X_i). Then, using the definition of conditional mutual information, we rewrite as follows:

$$\begin{aligned} I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i X_{i-1}) &= \mathop{\mathbb{E}}_{x_{i-1} \leftarrow X_{i-1}} \mathop{\mathbb{E}}_{x_i \leftarrow X_i} I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i = x_i, X_{i-1} = x_{i-1}) \\ &= \mathop{\mathbb{E}}_{x_{i-1} \leftarrow X_{i-1}} \mathop{\mathbb{E}}_{x_i \leftarrow X_i} I(M_{i-1} \ : \ Y_{i-1}^{x_i} \mid X_{i-1} = x_{i-1}) \ . \end{aligned} \tag{10}$$

Let $\mathcal{X}(x_{i-1})$ be the set of all subsets of x_{i-1} of size n_i. Observe that the distribution of X_i is uniform among the elements $\mathcal{X}(x_{i-1})$. Next, since $n_i \mid n_{i+1}$, by Baranyai's theorem [8] (see Theorem 4.4), the set $\mathcal{X}(x_{i-1})$ can be partitioned into $|\mathcal{X}(x_{i-1})|\frac{n_i}{n_{i-1}}$ subsets $\mathcal{X}_1(x_{i-1}), \mathcal{X}_2(x_{i-1}), \ldots$ such that $\cup_{x \in \mathcal{X}_i(x_{i-1})} x = x_{i-1}$. Denote the elements of set $\mathcal{X}_j(x_{i-1})$ by $x_j^1, x_j^2, \ldots, x_j^{\frac{n_{i-1}}{n_i}}$. We thus have:

$$\begin{aligned} &\mathop{\mathbb{E}}_{x_i \leftarrow X_i} I(M_{i-1} \ : \ Y_{i-1}^{x_i} \mid X_{i-1} = x_{i-1}) \\ &\quad = \frac{1}{|\mathcal{X}(x_{i-1})|} \sum_{x_i \in \mathcal{X}(x_{i-1})} I(M_{i-1} \ : \ Y_{i-1}^{x_i} \mid X_{i-1} = x_{i-1}) \\ &\quad = \frac{1}{|\mathcal{X}(x_{i-1})|} \sum_{j \in [|\mathcal{X}(x_{i-1})|\frac{n_i}{n_{i-1}}]} \sum_{\ell \in [\frac{n_{i-1}}{n_i}]} I(M_{i-1} \ : \ Y_{i-1}^{x_j^\ell} \mid X_{i-1} = x_{i-1}) \\ &\quad \leq \frac{1}{|\mathcal{X}(x_{i-1})|} \sum_{j \in [|\mathcal{X}(x_{i-1})|\frac{n_i}{n_{i-1}}]} \sum_{\ell \in [\frac{n_{i-1}}{n_i}]} I(M_{i-1} \ : \ Y_{i-1}^{x_j^\ell} \mid Y_{i-1}^{x_j^1} \ldots Y_{i-1}^{x_j^{\ell-1}}, X_{i-1} = x_{i-1}) \\ &\quad = \frac{1}{|\mathcal{X}(x_{i-1})|} \sum_{j \in [|\mathcal{X}(x_{i-1})|\frac{n_i}{n_{i-1}}]} I(M_{i-1} \ : \ Y_{i-1} \mid X_{i-1} = x_{i-1}) \\ &\quad = \frac{n_i}{n_{i-1}} I(M_{i-1} \ : \ Y_{i-1} \mid X_{i-1} = x_{i-1}) \ , \end{aligned} \tag{11}$$

where we used Lemma 4.2 to obtain the first inequality and the chain rule for mutual information for the subsequent equality. Combining Inequalities 10 and 11, we obtain:

$$\begin{aligned} I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i X_{i-1}) &\leq \mathop{\mathbb{E}}_{x_{i-1} \leftarrow X_{i-1}} \frac{n_i}{n_{i-1}} I(M_{i-1} \ : \ Y_{i-1} \mid X_{i-1} = x_{i-1}) \\ &= \frac{n_i}{n_{i-1}} I(M_{i-1} \ : \ Y_{i-1} \mid X_{i-1}) \leq \frac{n_i}{n_{i-1}} H(M_{i-1}) \leq \frac{n_i}{n_{i-1}} |M_{i-1}| \ . \end{aligned}$$

□

The next lemma shows that the last party's knowledge about the crucial bits $Y_1^{X_2}, Y_2^{X_3}, \ldots, Y_{p-1}^{X_p}$ is limited.

Lemma 4.6. *The following inequality holds: (recall that $s = \max_i |M_i|$)*

$$I(M_{p-1} X_p Y_p \ : \ Y_1^{X_2} Y_2^{X_3} \ldots Y_{p-1}^{X_p}) \leq \frac{s(p-1)}{n^{\frac{1}{p-1}}} \ .$$

Proof. Let $3 \leq i \leq p$ be an integer. Then:

$$\begin{aligned} I(M_{i-1} X_i Y_i \ : \ Y_1^{X_2} \ldots Y_{i-1}^{X_i}) &= I(X_i Y_i \ : \ Y_1^{X_2} \ldots Y_{i-1}^{X_i}) + I(M_{i-1} \ : \ Y_1^{X_2} \ldots Y_{i-1}^{X_i} \mid X_i Y_i) \\ &= 0 + I(M_{i-1} \ : \ Y_1^{X_2} \ldots Y_{i-1}^{X_i} \mid X_i) \ , \end{aligned} \tag{12}$$

where we first applied the chain rule, then used that the respective random variables are independent, and finally eliminated the conditioning on Y_i, which can be done since all other variables are independent with Y_i (see Rule 5 in Section 4.2). Next, we apply the chain rule again, invoke Lemma 4.5, and remove variables from the conditioning as they are independent with all other variables:

$$\begin{aligned} I(M_{i-1} \ : \ Y_1^{X_2} \ldots Y_{i-1}^{X_i} \mid X_i) &= I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i) + I(M_{i-1} \ : \ Y_1^{X_2} \ldots Y_{i-2}^{X_{i-1}} \mid X_i Y_{i-1}^{X_i}) \\ &\leq |M_{i-1}| \frac{n_p}{n_{p-1}} + I(M_{i-1} \ : \ Y_1^{X_2} \ldots Y_{i-2}^{X_{i-1}} \mid Y_{i-1}^{X_i}) \ . \end{aligned}$$

Next, we bound the term $I(M_{i-1} \ : \ Y_1^{X_2} \ldots Y_{i-2}^{X_{i-1}} \mid Y_{i-1}^{X_i})$ by using the data processing inequality, the chain rule, and remove an independent variable from the conditioning:

$$\begin{aligned} I(M_{i-1} \ : \ Y_1^{X_2} \ldots Y_{i-2}^{X_{i-1}} \mid Y_{i-1}^{X_i}) &\leq I(M_{i-2} X_{i-1} Y_{i-1} \ : \ Y_1^{X_2} \ldots Y_{i-2}^{X_{i-1}} \mid Y_{i-1}^{X_i}) \\ &= I(X_{i-1} Y_{i-1} \ : \ Y_1^{X_2} \ldots Y_{i-2}^{X_{i-1}} \mid Y_{i-1}^{X_i}) \\ &\quad + I(M_{i-2} \ : \ Y_1^{X_2} \ldots Y_{i-2}^{X_{i-1}} \mid X_{i-1} Y_{i-1} Y_{i-1}^{X_i}) \\ &= 0 + I(M_{i-2} \ : \ Y_1^{X_2} \ldots Y_{i-2}^{X_{i-1}} \mid X_{i-1}) \ . \end{aligned}$$

We have thus shown:

$$I(M_{i-1} \ : \ Y_1^{X_2} \ldots Y_{i-1}^{X_i} \mid X_i) \leq |M_{i-1}| \frac{n_i}{n_{i-1}} + I(M_{i-2} \ : \ Y_1^{X_2} \ldots Y_{i-2}^{X_{i-1}} \mid X_{i-1}) \ . \tag{13}$$

Using a slightly simpler version of the same reasoning, we can show that:

$$I(M_1 \ : \ Y_1^{X_2} \mid X_2) \leq |M_1| \frac{n_2}{n_1} \ . \tag{14}$$

Using Equality 12 and Inequalities 13 and 14, we obtain:

$$I(M_{p-1} \ : \ Y_1^{X_2} \dots Y_{p-1}^{X_p} X_p Y_p) \ \leq \ s\left(\frac{n_p}{n_{p-1}} + \frac{n_{p-1}}{n_{p-2}} + \dots + \frac{n_2}{n_1}\right) = \frac{(p-1)s}{n^{\frac{1}{p-1}}} \ .$$

□

Finally we are ready to proof the main result of this section.

Theorem 4.7. *For every $\epsilon < 0.005$, the randomized one-way communication complexity of Bit-Vector Learning(p, n, k) is bounded as follows:*

$$R_\epsilon^{\rightarrow}(\textit{Bit-Vector Learning}(p, n, k)) \geq \frac{(0.005k - 1)n^{\frac{1}{p-1}}}{p-1} = \Omega(\frac{kn^{\frac{1}{p-1}}}{p}) \ .$$

Proof. Let q be the largest integer i such that $Y_i^I \neq \epsilon$. Recall that by Lemma 4.3 we have $I(M_{p-1}X_pY_p \ : \ Z^I) \geq (1-\epsilon)1.01k - 1$. However, we also obtain:

$$\begin{aligned} I(M_{p-1}X_pY_p \ : \ Z^I) &= I(M_{p-1}X_pY_p \ : \ Y_1^I Y_2^I \dots Y_q^I) \\ &= I(M_{p-1}X_pY_p \ : \ Y_1^I Y_2^I \dots Y_{q-1}^I) + I(M_{p-1}X_pY_p \ : \ Y_q^I \mid Y_1^I Y_2^I \dots Y_{q-1}^I) \\ &\leq I(M_{p-1}X_pY_p \ : \ Y_1^I Y_2^I \dots Y_{q-1}^I) + H(Y_q^I) \\ &\leq I(M_{p-1}X_pY_p \ : \ Y_1^{X_2} Y_2^{X_3} \dots Y_{p-1}^{X_p}) + k \leq \frac{(p-1)s}{n^{\frac{1}{p-1}}} + k \ , \end{aligned}$$

where we first applied the chain rule for mutual information, then observed that the variables $Y_1^I Y_2^I \dots Y_{q-1}^I$ are contained in the variables $Y_1^{X^2} Y_2^{X^3} \dots Y_{p-1}^{X_p}$, and then invoked Lemma 4.6. This is thus only possible if:

$$(1-\epsilon)1.01k - 1 \leq \frac{(p-1)s}{n^{\frac{1}{p-1}}} + k \ ,$$

which, using $\epsilon < 0.005$, implies

$$\frac{(0.005k - 1)n^{\frac{1}{p-1}}}{p-1} \leq s \ .$$

Since we considered an arbitrary protocol Π, the result follows. □

4.5 Reduction: Neighborhood Detection to Bit-Vector Learning

In this subsection, we show that a streaming algorithm for Neighborhood Detection implies a communication protocol for Bit-Vector Learning. The lower bound on the communication complexity of Bit-Vector Learning thus yields a lower bound on the space requirements of any algorithm for Neighborhood Detection.

Theorem 4.8. *Let* $\mathbf{A}$ *be a c-approximation streaming algorithm for Neighborhood Detection with error probability at most 0.005 and $c = \frac{p}{1.01}$, for some integer $p \geq 2$. Then* $\mathbf{A}$ *uses space at least:*

$$\Omega(\frac{dn^{\frac{1}{p-1}}}{c^2}) \ .$$

Proof. Given their inputs for Bit-Vector Learning(p, n, k), the p parties construct a graph

$$G = ([n], [2kp], \cup_{i=1}^{p} E_i)$$

so that party i holds edges E_i. The edges of party $i \in [p]$ are as follows:

$$E_i = \{(\ell, 2k \cdot (i-1) + 2 \cdot (j-1) + Y_i^{\ell}[j] + 1) \ : \ \ell \in X_i \text{ and } j \in [k]\} \ .$$

Observe that $\Delta = kp$ (the vertex in X_p has such a degree).

Let $\mathbf{A}$ be a c-approximation streaming algorithm for Neighborhood Detection(n, d) with $c = \frac{p}{1.01}$ and $d = \Delta = kp$. Party 1 simulates algorithm $\mathbf{A}$ on their edges E_1 and sends the resulting memory state to party 2. This continues until party p completes algorithm and outputs a neighborhood (I, S). We observe that every neighbor $s \in S$ of vertex I allows us to determine one bit of string Z^I. Since the approximation factor of $\mathbf{A}$ is $\frac{p}{1.01}$, we have $|S| \geq \frac{1.01 \cdot \Delta}{p} = 1.01k$. We can thus predict $1.01k$ bits of string Z^I. By Theorem 4.7, every such protocol requires a message of length

$$\Omega(\frac{kn^{\frac{1}{p-1}}}{p}) = \Omega(\frac{dn^{\frac{1}{p-1}}}{c^2}) \ ,$$

which implies the same space lower bound for $\mathbf{A}$. □

5 Upper Bound for Insertion-deletion Streams

In this section, we discuss our streaming algorithm for Neighborhood Detection for insertion-deletion streams.

Our algorithm is based on the combination of two sampling strategies which both rely on the very common l_0-*sampling* technique: An l_0-sampler in insertion-deletion streams outputs a uniform random element from the non-zero coordinates of the vector described by the input stream. In our setting, the input vector is of dimension $n \cdot m$ where each coordinate indicates the presence or absence of an edge. Jowhari et al. showed that there is an l_0-sampler that uses space $O(\log^2(dim) \log \frac{1}{\delta})$, where dim is the dimension of the input vector, and succeeds with probability $1 - \delta$ [32].

In the following, we will run $\tilde{\mathrm{O}}(nd)$ l_0-samplers. To ensure that they succeed with large enough probability, we will run those samplers with $\delta = \frac{1}{n^{10}d}$ which yields a space requirement of $O(\log^2(nm) \cdot \log(nd))$ for each sampler.

l_0-sampling allows us to, for example, sample uniformly at random from all edges of the input graph or from all edges incident to a specific vertex.

Our algorithm is as follows:

1. Let $x = \max\{\frac{n}{c}, \sqrt{n}\}$
2. **Vertex Sampling:** Before processing the stream, sample a uniform random subset $A' \subseteq A$ of size $10x \ln n$. For each sampled vertex a, run $10\frac{d}{c} \ln n$ l_0-samplers on the set of edges incident to a. This strategy requires space $\tilde{\mathrm{O}}(\frac{xd}{c})$.
3. **Edge Sampling:** Run $10\frac{nd}{c}\left(\frac{1}{x} + \frac{1}{c}\right) \ln(nm)$ l_0-samplers on the stream, each producing a uniform random edge. This strategy requires space $\tilde{\mathrm{O}}\left(\frac{nd}{c}\left(\frac{1}{x} + \frac{1}{c}\right)\right)$.
4. Output any neighborhood of size at least $\frac{d}{c}$ among the stored edges if there is one, otherwise report `fail`

Algorithm 3: One-pass streaming algorithm for insertion-deletion streams

The analysis of our algorithm relies on the following lemma, whose proof uses standard concentration bounds and is deferred to the appendix.

Lemma 5.1. *Let y, k, n be integers with $y \leq k \leq n$. Let $\mathcal{U}$ be a universe of size n and let $X \subseteq \mathcal{U}$ be a subset of size k. Further, let Y be the subset of $\mathcal{U}$ obtained by sampling $C \ln(n)\frac{ny}{k}$ times from $\mathcal{U}$ uniformly at random (with repetition), for some $C \geq 4$. Then, $|Y \cap X| \geq y$ with probability $1 - \frac{1}{n^{C-3}}$.*

We will first show that if the input graph contains enough vertices of degree at least $\frac{d}{c}$, then the vertex sampling strategy succeeds.

Lemma 5.2. *The vertex sampling strategy succeeds with high probability if there are at least $\frac{n}{x}$ vertices of degree at least $\frac{d}{c}$.*

Proof. First, we show that A' contains a vertex of degree at least $\frac{d}{c}$ with high probability. Indeed, the probability that no node of degree at least $\frac{d}{c}$ is contained in the sample A' is at most:

$$\frac{\binom{n-\frac{n}{x}}{10x\ln n}}{\binom{n}{10x\ln n}} = \frac{(n-\frac{n}{x})!\cdot(n-10x\ln n)!}{n!\cdot(n-\frac{n}{x}-10x\ln n)!} \leq \left(\frac{n-10x\ln n}{n}\right)^{\frac{n}{x}} \leq \exp\left(-\frac{10x\ln n}{n}\cdot\frac{n}{x}\right) = n^{-10} .$$

Next, suppose that there is a node $a \in A'$ with $\deg(a) \geq \frac{d}{c}$. Then, by Lemma 5.1 sampling $10\cdot\frac{d}{c}\log n$ times uniformly at random from the set of edges incident to a results in at least $\frac{d}{c}$ different edges with probability at least $1 - n^{-7}$. □

Next, we will show that if the vertex sampling strategy fails, then the edge sampling strategy succeeds.

Lemma 5.3. *The edge sampling strategy succeeds with high probability if there are at most $\frac{n}{x}$ vertices of degree at least $\frac{d}{c}$.*

Proof. Let Δ be the largest degree of an A-vertex. Since there are at most $\frac{n}{x}$ A-vertices of degree at least $\frac{d}{c}$, the input graph has at most $|E| \leq \frac{n}{x}\cdot\Delta + n\cdot\frac{d}{c}$ edges. Fix now a node a of degree Δ. Then, by Lemma 5.1, we will sample $\frac{d}{c}$ edges incident to a with high probability, if we sample

$$10\cdot\frac{|E|\frac{d}{c}}{\Delta}\ln(|E|) \leq 10\cdot\left(\frac{nd}{xc}+\frac{nd^2}{c^2\Delta}\right)\ln(|E|) \leq 10\cdot\frac{nd}{c}\left(\frac{1}{x}+\frac{1}{c}\right)\ln(n\cdot m) ,$$

which matches the number of samples we take in our algorithm. □

We obtain the following theorem:

Theorem 5.4. *Algorithm 3 is a one-pass c-approximation streaming for insertion-deletion streams that uses space $\tilde{\mathrm{O}}(\frac{dn}{c^2})$ if $c \leq \sqrt{n}$, and space $\tilde{\mathrm{O}}(\frac{\sqrt{n}d}{c})$ if $c > \sqrt{n}$, and succeeds with high probability.*

Proof. Correctness of the algorithm follows from Lemmas 5.2 and 5.3. Concerning the space requirements, the algorithm uses space $\tilde{\mathrm{O}}(\frac{xd}{c}) + \tilde{\mathrm{O}}(\frac{nd}{c}(\frac{1}{x}+\frac{1}{c}))$, which simplifies to the bounds claimed in the statement of the theorem by choosing $x = \max\{\frac{n}{c}, \sqrt{n}\}$. □

Using the same technique as in the proof of Corollary 3.3, we obtain:

Corollary 5.5. *There is a $\mathrm{O}(\sqrt{n})$-approximation semi-streaming algorithm for insertion-deletion streams for* Star Detection *that succeeds with high probability.*

6 Lower Bound for Insertion-deletion Streams

We will give now our lower bound for Neighborhood Detection in insertion-deletion streams. To this end, we first define a two-party communication problem denoted Augmented-Matrix-Row-Index and then lower bound its communication complexity. Finally, we argue that an insertion-deletion streaming algorithm for Neighborhood Detection can be used to solve Augmented-Matrix-Row-Index, which yields the desired lower bound.

6.1 The Augmented-Matrix-Row-Index Problem

Before defining the problem of interest, we require additional notation. Let M be an n-by-m matrix. Then the ith row of M is denoted M_i. A position (i, j) is a tuple chosen from $[n] \times [m]$. We will index the matrix M by a set of positions S, i.e., M_S, meaning the matrix positions $M_{i,j}$, for every $(i, j) \in S$.

The problem Augmented-Matrix-Row-Index(n, m, k) is defined as follows:

Problem 5 (Augmented-Matrix-Row-Index(n, m, k))**.** *In Augmented-Matrix-Row-Index, Alice holds a binary matrix $X \in \{0, 1\}^{n \times m}$ where every X_{ij} is a uniform random Bernoulli variable, for some integers n, m. Bob holds a uniform random index $J \in [n]$ and for each $i \neq J$, Bob holds a uniform random subset of positions $Y_i \subseteq \{i\} \times [m]$ with $|Y_i| = m - k$ and also knows X_{Y_i}. Alice sends a message to Bob who then outputs the entire row X_J.*

For ease of notation, we define $Y_I = \perp$ and $Y = Y_1, Y_2, \ldots, Y_n$. An example instance of Augmented-Matrix-Row-Index$(4, 6, 2)$ is given in Figure 3.

6.2 Lower Bound Proof for Augmented-Matrix-Row-Index

We now prove a lower bound on the one-way communication complexity of Augmented-Matrix-Row-Index(n, m, k). To this end, let Π be a deterministic communication protocol for Augmented-Matrix-Row-Index(n, m, k) with distributional error at most $\epsilon > 0$ and denote by M the message that Alice sends to Bob.

First, we prove that the mutual information between row X_J and Bob's knowledge, that is $MJYX_Y$, is large. Since the proof of the next lemma is almost identical to Lemma 4.3 we postpone it to the appendix:

Lemma 6.1. *We have:*

$$I(X_J \ : \ MJYX_Y) \geq (1 - \epsilon)m - 1 \ .$$

Next, we prove our communication lower bound for Augmented-Matrix-Row-Index:

Theorem 6.2. *We have:*

$$R^{\rightarrow}_{\epsilon}(\textit{Augmented-Matrix-Row-Index}(n, m, k)) \geq (n - 1)(k - 1 - \epsilon m) \ .$$

Proof. Our goal is to bound the term $I(X \ : \ M)$ from below. To this end, we partition the matrix M as follows: Let Z be all positions that are different to row J and the positions known to Bob, i.e., the set Y. Then:

$$I(X \ : \ M) \ = \ I(X_Y X_J X_Z \ : \ M) = I(X_Y X_J \ : \ M) + I(X_Z \ : \ M \mid X_J X_Y)$$

$$\geq \; I(X_Z \; : \; M \mid X_J X_Y) \; ,$$

where we applied the chain rule for mutual information. For $i \neq J$, let $Z_i = (\{i\} \times [m]) \setminus Y_i$, i.e., the positions of row i unknown to Bob, and let $Z_J = \perp$. Furthermore, let L be a random variable that is uniformly distributed in $[n] \setminus J$. Then, using the chain rule for mutual information and the fact that X_{Z_i} and X_{Z_j} are independent, for every $i \neq j$, we obtain:

$$\begin{aligned} I(X_Z \; : \; M \mid X_J X_Y) &= \sum_{i \in [n] \setminus J} I(X_{Z_i} \; : \; M| \; X_J X_Y X_{Z_1}, \ldots, X_{Z_{i-1}}) \\ &\geq \sum_{i \in [n] \setminus J} I(X_{Z_i} \; : \; M| \; X_J X_Y) \\ &= (n-1) \cdot I(X_{Z_L} \; : \; M| \; X_J X_Y L) \; . \end{aligned}$$

Our goal is to show that $I(X_{Z_L} \; : \; M| \; X_J X_Y L) \geq k$, which then completes the theorem. To this end, we will relate the previous expression to the statement in Lemma 6.1, as follows: First, let Y'_J be $m-k$ uniform random positions in row J. Then by independence, we obtain

$$I(X_{Z_L} \; : \; M| \; X_J X_Y L) \geq I(X_{Z_L} \; : \; M| \; X_{Y'_J} X_Y L) \; .$$

Next, denote by $Y \setminus Y_L := Y_1, \ldots, Y_{L-1}, Y_{L+1}, \ldots, Y_n$. Then, by using the chain rule again, we obtain:

$$\begin{aligned} I(X_L \; : \; M| \; X_{Y'_J} X_{Y \setminus Y_L} L) &= I(X_{Y_L} X_{Z_L} \; : \; M| \; X_{Y'_J} X_{Y \setminus Y_L} L) \\ &= I(X_{Y_L} \; : \; M| \; X_{Y'_J} X_{Y \setminus Y_L} L) + I(X_{Z_L} \; : \; M| \; X_{Y'_J} X_Y L) \\ &\leq H(X_{Y_L}) + I(X_{Z_L} \; : \; M| \; X_{Y'_J} X_Y L) \\ &\leq (m-k) + I(X_{Z_L} \; : \; M| \; X_{Y'_J} X_Y L) \; . \end{aligned}$$

Last, it remains to argue that $I(X_{Z_L} \; : \; M| \; X_{Y'_J} X_{Y \setminus Y_L} L)$ is equivalent to $I(X_{Z_J} \; : \; M| \; JY X_Y)$. Indeed, first observe that L is chosen uniformly at random from $[n] \setminus J$, which is equivalent to a value chosen uniformly at random from $[n]$ since J is itself a uniform random value in $[n]$. Observe further that the conditioning is also equivalent: both $X_{Y'_J} X_{Y \setminus Y_L}$ and X_Y reveal $m-k$ uniform random positions of each row different to row L and J, respectively. Hence, using Lemma 6.1 we obtain:

$$I(X_{Z_L} \; : \; M| \; X_{Y'_J} X_Y L) \geq I(X_L \; : \; M| \; X_{Y'_J} X_{Y \setminus Y_L} L) - (m-k) = (1-\epsilon)m - 1 - (m-k) = k - 1 - \epsilon m \; .$$

We have thus shown that $I(X \; : \; M) \geq (n-1)(k-1-\epsilon m)$. The result then follows, since $I(X \; : \; M) \leq H(M) \leq |M|$. □

6.3 Reduction: Neighborhood-Detection to Augmented-Matrix-Row-Index

Lemma 6.3. *Let $\boldsymbol{A}$ be a c-approximation insertion-deletion streaming algorithm for Neighborhood Detection(n, d) with space s that fails with probability at most δ. Then there is a one-way communication protocol for Augmented-Matrix-Row-Index$(n, 2d, \frac{d}{c} - 1)$ with message size*

$$O(s \cdot c \cdot \log n)$$

that fails with probability at most $\delta + n^{-10}$.

Proof. We will show how to solve Augmented-Matrix-Row-Index$(n, 2d, \frac{d}{c} - 1)$ using algorithm $\mathbf{A}$. Assume from now on that the number of 1s in row J of matrix X is at least d. We will argue later what to do if this is not the case. Alice and Bob repeat the following protocol $\Theta(c \log n)$ times in parallel:

First, Alice and Bob use public randomness to chose n permutations $\pi_i : [2d] \to [2d]$ at random and permute the elements of each row i independently using π_i. Observe that this operation does not change the number of 1s in each row. Let X' be the permuted matrix. Then, Alice and Bob interpret the matrix X' as the adjacency matrix of a bipartite graph, where Bob's knowledge about X' is treated as edge deletions. Under the assumption that row J contains at least d 1s, and since none of the elements of row J are deleted by Bob's input, we have a valid instance for Neighborhood Detection(n, d). Alice then runs $\mathbf{A}$ on the graph obtained from X' and sends the resulting memory state to Bob. Bob then continues $\mathbf{A}$ on his input and outputs a neighborhood of size at least $\frac{d}{c}$. Observe that after Bob's deletions, every row except row J contains at most $\frac{d}{c} - 1$ 1s, which implies that $\mathbf{A}$ reports a neighborhood rooted at A-vertex J (the vertex that corresponds to row J). Bob thus learns at least $\frac{d}{c}$ positions of row J where the matrix X' is 1. Bob then applies $(\pi_J)^{-1}$ and thus learns at least $\frac{d}{c}$ positions of row J of matrix X where the value is 1. Observe that since the permutation π_J was chosen uniformly at random, the probability that a specific position with value 1 in row J of matrix X is learnt by the algorithm is at least $\frac{d/c}{2d} = \frac{1}{2c}$. Applying concentration bounds, since the protocol is repeated $\Theta(c \cdot \log n)$ times (where Θ hides a large enough constant), we learn all 1s in row J with probability $1 - n^{-10}$ and thus have solved Augmented-Matrix-Row-Index$(n, 2d, \frac{d}{c} - 1)$.

It remains to address the case when row J contains fewer than d 1s. To address this case, Alice and Bob simultaneously run the algorithm mentioned above on the matrix obtained by inverting every bit, which allows them to learn all positions in row J where the matrix X is 0. Finally, Bob can easily decide in which of the two cases they are: If row J contained at most $d - 1$ 1s then the strategy without inverting the input would therefore report at most $d - 1$ 1s. Bob thus knows in which case they are. □

Theorem 6.4. *Every c-approximation insertion-deletion streaming algorithm for Neighborhood Detection(n, d) that fails with probability $\delta \leq \frac{1}{2d}$ requires space $\Omega\left(\frac{nd}{c^2 \log n})\right)$.*

Proof. Let $\mathbf{A}$ be a streaming algorithm as in the description of this theorem. Then, by Lemma 6.3, there is a one-way communication protocol for Augmented-Matrix-Row-Index$(n, 2d, \frac{d}{c} - 1)$ that succeeds with probability $\delta + n^{-10}$ and communicates $\mathrm{O}(s \cdot c \log n)$ bits. Then, by Theorem 6.2, we have:

$$s \cdot c \log n = \Omega\left((n-1)(\frac{d}{c} - 2 - (\delta + n^{-10})2d\right) = \Omega\left(\frac{nd}{c}\right) \ ,$$

which yields

$$s = \Omega\left(\frac{nd}{c^2 \log n}\right) \ .$$

□

References

[1] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete*

Algorithms, SODA '12, pages 459–467, Philadelphia, PA, USA, 2012. Society for Industrial and Applied Mathematics.

[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.

[3] Sepehr Assadi. Tight space-approximation tradeoff for the multi-pass streaming set cover problem. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 321–335, 2017.

[4] Sepehr Assadi, Sanjeev Khanna, and Yang Li. Tight bounds for single-pass streaming complexity of the set cover problem. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 698–711, 2016.

[5] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1345–1364, 2016.

[6] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. Information theory methods in communication complexity. In *Proceedings 17th IEEE Annual Conference on Computational Complexity*, pages 93–102, May 2002.

[7] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS '02, pages 209–218, Washington, DC, USA, 2002. IEEE Computer Society.

[8] Zsolt Baranyai. The edge-coloring of complete hypergraphs i. *Journal of Combinatorial Theory, Series B*, 26(3):276 – 294, 1979.

[9] Radu Berinde, Graham Cormode, Piotr Indyk, and Martin J. Strauss. Space-optimal heavy hitters with strong error bounds. In *Proceedings of the Twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '09, pages 157–166, New York, NY, USA, 2009. ACM.

[10] Arnab Bhattacharyya, Palash Dey, and David P. Woodruff. An optimal algorithm for ℓ1-heavy hitters in insertion streams and related problems. *ACM Trans. Algorithms*, 15(1):2:1–2:27, October 2018.

[11] Yousra Chabchoub, Christine Fricker, and Hanene Mohamed. Analysis of a bloom filter algorithm via the supermarket model. In *21st International Teletraffic Congress, ITC 2009, Paris, France, September 15-17, 2009*, pages 1–8, 2009.

[12] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*, pages 107–117, 2003.

[13] Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew Chi-Chih Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 270–278, 2001.

[14] Amit Chakrabarti and Anthony Wirth. Incidence geometries and the pass complexity of semi-streaming set cover. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1365–1373, 2016.

[15] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, ICALP '02, pages 693–703, Berlin, Heidelberg, 2002. Springer-Verlag.

[16] 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.

[17] Graham Cormode and Donatella Firmani. A unifying framework for ℓ0-sampling algorithms. *Distrib. Parallel Databases*, 32(3):315–335, September 2014.

[18] Graham Cormode and Hossein Jowhari. Lp samplers and their applications: A survey. *ACM Comput. Surv.*, 52(1):16:1–16:31, February 2019.

[19] Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, April 2005.

[20] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, New York, NY, USA, 2006.

[21] Erik D. Demaine, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. On streaming and communication complexity of the set cover problem. In *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*, pages 484–498, 2014.

[22] Yuval Emek and Adi Rosén. Semi-streaming set cover. *ACM Trans. Algorithms*, 13(1):6:1–6:22, 2016.

[23] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Trans. Comput. Syst.*, 21(3):270–313, August 2003.

[24] Shir Landau Feibish, Yehuda Afek, Anat Bremler-Barr, Edith Cohen, and Michal Shagam. Mitigating DNS random subdomain ddos attacks by distinct heavy hitters sketches. In *Proceedings of the fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies, HotWeb 2017, San Jose / Silicon Valley, CA, USA, October 12 - 14, 2017*, pages 8:1–8:6, 2017.

[25] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the streaming model: The value of space. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 745–754, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[26] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2):207 – 216, 2005. Automata, Languages and Programming: Algorithms and Complexity (ICALP-A 2004).

[27] Magnús M. Halldórsson, Xiaoming Sun, Mario Szegedy, and Chengu Wang. Streaming and communication complexity of clique approximation. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, pages 449–460, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[28] Sariel Har-Peled, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. Towards tight bounds for the streaming set cover problem. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 371–383, 2016.

[29] Monika R. Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. External memory algorithms. chapter Computing on Data Streams, pages 107–118. American Mathematical Society, Boston, MA, USA, 1999.

[30] Kaave Hosseini, Shachar Lovett, and Grigory Yaroslavtsev. Optimality of linear sketching under modular updates. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA.*, pages 13:1–13:17, 2019.

[31] Piotr Indyk, Sepideh Mahabadi, Ronitt Rubinfeld, Jonathan Ullman, Ali Vakilian, and Anak Yodpinyanee. Fractional set cover in the streaming model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, pages 12:1–12:20, 2017.

[32] 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. ACM.

[33] Christian Konrad. Maximum matching in turnstile streams. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015*, pages 840–852, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[34] Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In Anupam Gupta, Klaus Jansen, José Rolim, and Rocco Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 231–242, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[35] Abhishek Kumar and Jun (Jim) Xu. Sketch guided sampling - using on-line estimates of flow size for adaptive data collection. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 23-29 April 2006, Barcelona, Catalunya, Spain*, 2006.

[36] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 2006.

[37] Yi Li, Huy L. Nguyen, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 174–183, New York, NY, USA, 2014. ACM.

[38] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 346–357. VLDB Endowment, 2002.

[39] Andrew McGregor. Graph stream algorithms: A survey. *SIGMOD Rec.*, 43(1):9–20, May 2014.

[40] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proceedings of the 10th International Conference on Database Theory*, ICDT'05, pages 398–412, Berlin, Heidelberg, 2005. Springer-Verlag.

[41] Jayadev Misra and David Gries. Finding repeated elements. *Sci. Comput. Program.*, 2(2):143–152, 1982.

[42] Xiaoming Sun and David P. Woodruff. Tight bounds for graph problems in insertion streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, pages 435–448, 2015.

[43] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, March 1985.

A Sampling Lemma

Lemma 5.1. *Let y, k, n be integers with $y \leq k \leq n$. Let $\mathcal{U}$ be a universe of size n and let $X \subseteq \mathcal{U}$ be a subset of size k. Further, let Y be the subset of $\mathcal{U}$ obtained by sampling $C \ln(n) \frac{ny}{k}$ times from $\mathcal{U}$ uniformly at random (with repetition), for some $C \geq 4$. Then, $|Y \cap X| \geq y$ with probability $1 - \frac{1}{n^{C-3}}$.*

Proof. Let t_i be the expected number of samples it takes to sample an item from X that has not been sampled previously, given that $i-1$ items from X have already been sampled. The probability of sampling a new item given that $i-1$ items have already been sampled is $p_i = \frac{k-(i-1)}{n}$, which implies that $t_i = \frac{1}{p_i} = \frac{n}{k-(i-1)}$. Thus, the expected number μ of samples required to sample at least y different items is therefore:

$$\mu := \sum_{i=1}^{y} t_i = \sum_{i=1}^{y} \frac{n}{k-(i-1)} = n \cdot (H_k - H_{k-y}) = n \cdot H \ ,$$

where H_i is the i-th Harmonic number and $H = H_k - H_{k-y}$. We consider two cases:

Suppose first that $y \geq \frac{k}{2}$. Then, we use the approximation $n \leq \mu \leq n\ln(k)$. By a Chernoff bound, the probability that more than $C\ln(n)\frac{ny}{k} \geq \frac{C}{2}n\ln(n)$ samples are needed is at most

$$\exp\left(-\frac{(\frac{C}{2}-1)^2}{2+\frac{C}{2}-1}n\right) \leq \exp\left(-\frac{1}{2}n\right) \ .$$

Next, suppose that $y < \frac{k}{2}$. Then, we use the (crude) approximations $1 \leq \mu \leq \frac{ny}{k}$. By a Chernoff bound, the probability that more than $C\ln(n)\frac{ny}{k}$ samples are needed is at most:

$$\exp\left(-\frac{(C-1)^2\ln(n)^2}{2+(C-1)\ln n}\right) \leq n^{-C+3} \ .$$

□

B Missing Proof: Insertion-deletion Stream Lower Bound

Lemma 6.1 *We have:*

$$I(X_J \ : \ MJYX_Y) \geq (1-\epsilon)m - 1 \ .$$

Proof. Let Out be the output produced by the protocol for Augmented-Matrix-Row-Index. We will first bound the term $I(Out \ : \ X_J) = H(X_J) - H(X_J \mid Out)$. To this end, let E be the indicator random variable of the event that the protocol errs. Then, $\mathbb{P}[E=1] \leq \epsilon$. We have:

$$H(E, X_J \mid Out) \ = \ H(X_J \mid Out) + H(E \mid Out, X_J) = H(X_J \mid Out) \ , \tag{15}$$

where we used the chain rule for entropy and the fact that $H(E \mid Out, X_J) = 0$ since E is fully determined by Out and X_J. Furthermore,

$$H(E, X_J \mid Out) \ = \ H(E \mid Out) + H(X_J \mid E, Out) \leq 1 + H(X_J \mid E, Out) \ , \tag{16}$$

using the chain rule for entropy and the bound $H(E \mid Out) \leq H(E) \leq 1$. From Inequalities 15 and 16 we obtain:

$$H(X_J \mid Out) \ \leq \ 1 + H(X_J \mid E, Out) \ . \tag{17}$$

Next, we bound the term $H(X_J \mid E, Out)$ as follows:

$$H(X_J \mid E, Out) \ = \ \mathbb{P}\left[E=0\right] H(X_J \mid Out, E=0) + \mathbb{P}\left[E=1\right] H(X_J \mid Out, E=1) \ . \tag{18}$$

Concerning the term $H(X_J \mid Out, E=0)$, since no error occurs, Out determines X_J. We thus have that $H(X_J \mid Out, E=0) = 0$. We bound the term $H(X_J \mid Out, E=1)$ by $H(X_J \mid Out, E=1) \leq H(X_J) = m$ (since conditioning can only decrease entropy). The quantity $H(X_J \mid E, Out)$ can thus be bounded as follows:

$$H(X_J \mid E, Out) \ \leq \ (1-\epsilon)\cdot 0 + \epsilon H(X_J) = \epsilon H(X_J) \ . \tag{19}$$

Next, using Inequalities 17 and 19, we thus obtain:

$$\begin{aligned} I(Out \ : \ X_J) \ &= \ H(X_J) - H(X_J \mid Out) \geq H(X_J) - 1 - H(X_J \mid E, Out) \\ &\geq \ H(X_J) - 1 - \epsilon H(X_J) = (1-\epsilon)H(X_J) - 1 = (1-\epsilon)m - 1 \ . \end{aligned}$$

Last, observe that Out is a function of M, J, Y and X_Y. The result then follows from the data processing inequality. □