Frequent Elements with Witnesses in Data Streams

Christian Konrad
christian.konrad@bristol.ac.uk
Department of Computer Science, University of Bristol
Bristol, United Kingdom

ABSTRACT

Detecting frequent elements is among the oldest and most-studied problems in the area of data streams. Given a stream of m data items in $\{1, 2, \ldots, n\}$, the objective is to output items that appear at least d times, for some threshold parameter d, and provably optimal algorithms are known today. However, in many applications, knowing only the frequent elements themselves is not enough: For example, an Internet router may not only need to know the most frequent destination IP addresses of forwarded packages, but also the timestamps of when these packages appeared or any other meta-data that "arrived" with the packages, e.g., their source IP addresses.

In this paper, we introduce the witness version of the frequent elements problem: Given a desired approximation guarantee $\alpha \geq 1$ and a desired frequency $d \leq \Delta$, where Δ is the frequency of the most frequent item, the objective is to report an item together with at least d/α timestamps of when the item appeared in the stream (or any other meta-data that arrived with the items). We give provably optimal algorithms for both the insertion-only and insertion-deletion stream settings: In insertion-only streams, we show that space $\tilde{O}(n + d \cdot n^{\frac{1}{\alpha}})$ is necessary and sufficient for every integral $1 \leq \alpha \leq \log n$. In insertion-deletion streams, we show that space $\tilde{O}(\frac{n \cdot d}{\alpha^2})$ is necessary and sufficient, for every $\alpha \leq \sqrt{n}$.

CCS CONCEPTS

• **Theory of computation → Streaming, sublinear and near linear time algorithms**; **Streaming models**.

KEYWORDS

frequent elements, heavy hitters, data streams, algorithms, lower bounds

ACM Reference Format:
Christian Konrad. 2018. Frequent Elements with Witnesses in Data Streams. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/1122445.1122456

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, 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.

The Frequent Elements (FE) (or heavy hitters) problem is among the oldest and most-studied problems in the area of data streams. Given a stream $S = s_1, s_2, \ldots, s_m$ of length m with $s_i \in [n]$, for some integer n, the goal is to identify items in $[n]$ that appear at least $d = \epsilon m$ times, for some $\epsilon > 0$. This problem was first solved by Misra and Gries in 1982 [37] and has since been addressed in countless research papers (e.g. [8, 10, 11, 14, 17, 21, 31, 33, 36]), culminating in provably optimal algorithms [10].

However, in many applications, only knowing the frequent items themselves is insufficient, and additional application-specific data is required. For example:

- Given a database log, a FE algorithm can be used to detect a frequently updated (or queried) entry. However, users who committed these updates (or queries) or the timestamps of when these updates (or queries) were executed cannot be reported by such an algorithm.
- Given a stream of friendship updates in a social network graph, a FE algorithm can detect nodes of large degree (e.g., an influencer in a social network). Their neighbours (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 frequent elements*, that is, frequent target IP addresses that are requested from many distinct sources [22]. Here, a (distinct) FE 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 source IP addresses from where the attacks originated remain unknown.

In this paper, we introduce the witness version of the frequent elements problem, which captures the examples mentioned above. This problem is formulated as a problem on graphs:

Problem 1 (Frequent Elements with Witnesses (FEwW). *In FEwW(n, d), the input consists of a bipartite graph $G = (A, B, E)$ with $|A| = n$ and $|B| = m = \text{poly}\, n$, and a threshold parameter d. We are given the promise that there is at least one A-vertex of degree at least d. The goal is to output an A-vertex together with at least d/α of its neighbours, for some approximation factor $\alpha \geq 1$.*

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/1122445.1122456

FEwW allows us to model frequent elements problems where, besides the frequent elements themselves, additional satellite data that "arrives" together with the input items also needs to be reported. For example, in the database log example above, database entries can be regarded as A-vertices, users as B-vertices, and updates/queries as edges connecting entries to users. The incident edges of a node reported by an algorithm for FEwW can be regarded as a "witness" that proves that the node is indeed of large degree. The restriction $|B| = \text{poly}\, n$ is only imposed for convenience as it is reasonable and simplifies the complexity bounds of our algorithms.

Our aim is to solve FEwW in two models of graph streams: In the *insertion-only* model, a streaming algorithm receives the edges of G in arbitrary order. In the *insertion-deletion* model, a streaming algorithm receives an arbitrary sequence of edge insertion and deletions. In both models, the objective is to design algorithms with minimal space.

Formulating FEwW as a graph problem has two advantages: First, it allows for the same satellite data of different input items. Second, a streaming algorithm for FEwW can be used to solve the related Star Detection problem, a subgraph detection problem that deserves attention in its own right:

Problem 2 (Star Detection). *In Star Detection, the input is a general graph $G = (V, E)$. The objective is to output the largest star in G, i.e., determining a node of largest degree together with its neighbourhood. An α-approximation algorithm ($\alpha \geq 1$) to Star Detection outputs a node together with at least Δ/α of its neighbours, where Δ is the maximum degree in the input graph.*

For example, Star Detection can be used to solve the second example mentioned above, i.e., finding influencers together with their followers in social networks.

1.1 Our Results

In this paper, we resolve the space complexity of streaming algorithms for FEwW in both insertion-only and insertion-deletion streams up to poly-logarithmic factors.

In insertion-only streams, we give an α-approximation streaming algorithm with space[1] $\tilde{O}(n + n^{\frac{1}{\alpha}} d)$ that succeeds with high probability[2], for integral values of $\alpha \geq 1$ (**Theorem 3.2**). We complement this result with a lower bound, showing that space $\Omega(n/\alpha^2 + n^{\frac{1}{\alpha-1}} d/\alpha^2)$ is necessary for every algorithm that computes a $\alpha/1.01$ approximation, for every integer $\alpha \geq 2$ (**Theorems 4.1 and 4.8**). Observe that the latter result also implies a lower bound of $\Omega(n/\alpha^2 + n^{\frac{1}{\alpha}} d/\alpha^2)$ for every α-approximation algorithm, where α is integral. Up to poly-logarithmic factors, our algorithm is thus optimal for every poly-logarithmic α.

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

Our lower bounds translate to Star Detection with parameter $d = \Theta(n)$, and our algorithms translate to Star Detection by setting $d = \Theta(n)$ in the space bound and by introducing an additional $\log_{1+\epsilon} n$ factor in the space complexities and a $1 + \epsilon$ factor in the approximation ratios (**Lemma 3.3**). For example, a $O(\log n)$-approximation to Star Detection can be computed in insertion-only streams in space $\tilde{O}(n)$ (graph streaming algorithms with space $\tilde{O}(n)$ are referred to as *semi-streaming algorithms* [23]), while such an approximation would require space $\tilde{\Omega}(n^2)$ in insertion-deletion streams.

1.2 Techniques

Our insertion-only streaming algorithm for FEwW makes use of a subroutine that solves the following sampling task: For degree bounds $d_1 < d_2$ and an integer s, compute a uniform random sample of size s of the A-vertices of degree at least d_1, and, for every sampled vertex $a \in A$, compute $\min\{d_2, \deg(a) - d_1 + 1\}$ incident edges to a. We say that this task *succeeds* if there is one sampled node for which d_2 incident edges are computed. We give a streaming algorithm, denoted Deg-Res-Sampling(d_1, d_2, s), that solves this task, using a combination of reservoir sampling [38] and degree counts. Next, we run α instances of Deg-Res-Sampling(d_1, d_2, s) in parallel, for changing parameter $d_1 = i \cdot \frac{d}{\alpha}$, for $i = 0, 1, \ldots, \alpha - 1$, and fixed parameters $d_2 = \frac{d}{\alpha}$ and $s = \tilde{\Theta}(n^{1/\alpha})$. It can be seen that run i succeeds if the ratio of the number of nodes of degree at least $i \cdot \frac{d}{\alpha}$ to the number of nodes of degree at least $(i + 1) \cdot \frac{d}{\alpha}$ in the input graph is not too large, i.e., in $O(n^{1/\alpha})$. We prove that this condition is necessarily fulfilled for at least one of the parallel runs.

Our lower bound for insertion-only streams is the most technical contribution of this paper. We show that a streaming algorithm for FEwW can be used to solve a new multi-party one-way communication problem denoted Bit-Vector-Learning, where the bits of multiple binary strings of different lengths are partitioned among multiple parties. The last party is required to output enough bits of at least one of the strings - this is difficult, since the partitioning is done so that not a single party alone holds enough bits of any of the strings. We prove a lower bound on the communication complexity of Bit-Vector-Learning via information theoretic arguments, which then translates to FEwW. A highlight of our technique is the application of Baranyai's theorem for colouring complete regular hypergraphs [7], which allows us to partition and subsequently quantify the information that is necessarily revealed when solving Bit-Vector-Learning.

FEwW is much harder to solve in insertion-deletion streams and requires a different set of techniques. Our insertion-deletion streaming algorithm employs two sampling strategies: A vertex-based sampling strategy that succeeds if the input graph is dense enough, and an edge sampling strategy that succeeds if the input graph is relatively sparse. We implement both sampling methods using l_0-sampling techniques [26].

Last, our lower bound for insertion-deletion streams is proved in the one-way two-party communication model and is conceptually interesting since it extends the traditional one-way two-party

[1] We use $\tilde{O}$, $\tilde{\Theta}$, and $\tilde{\Omega}$ to mean O, Θ, and Ω (respectively) with poly-log factors suppressed.

[2] 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.

Augmented-Index communication problem to a suitable two dimensional version that may be of independent interest. Similar to our lower bound for insertion-only streams, we use information-theoretic arguments to prove a tight lower bound.

1.3 Further Related Work

As previousy mentioned, the traditional (without witnesses) FE problem is very well studied, and many algorithms with different properties are known, including Misra-Gries [37] (see also [20, 28]), CountSketch [15], Count-min Sketch [17], multi-stage Bloom filters [11], and many others [9, 21, 31, 33, 35]. A crucial difference between the witness and the without witness versions is that the space complexities behave inversely with regards to the desired frequency threshold parameter $1 \le d \le m$: Most streaming algorithms for FE use space proportional to $\frac{m}{d}$ (intuitively, the more often an element appears in the stream, the easier it is to pick it up using sampling), while the space is trivially $\Omega(d/\alpha)$ for FEwW, since at least d/α witnesses need to be reported by the algorithm. In terms of techniques, the two versions therefore have a very different flavour, and the FEwW problem is perhaps closer in spirit to the literature on graph streaming algorithms than to the (without witnesses) frequent elements literature.

Graph streaming algorithms in the insertion-only model have been studied since more than 20 years [25], and this model is fairly well understood today (see [34] for an excellent survey). The first techniques for processing insertion-deletion graph streams were introduced in a seminal paper by Ahn et al. [1] in 2012. While many problems, such as Connectivity [1], Spectral Sparsification [27], and $(\Delta + 1)$-colouring [2], are known to be equally hard in both the insertion-only and the insertion-deletion settings (up to a poly-logarithmic factor difference in the space requirements), only few problems, such as Maximum Matching and Minimum Vertex Cover, are known to be substantially harder in the insertion-deletion setting [4, 19, 29]. In this paper, we prove that FEwW and Star Detection are much harder in the insertion-deletion setting than in the insertion-only setting, thereby establishing another separation result between the two models.

Star Detection shares similarities with other subgraph approximation problems, such as Maximum Independent Set/Maximum Clique [16, 24], Maximum Matching [4, 19, 23, 29, 30], and Minimum Vertex Cover [19], which can all be solved approximately using sublinear (in n^2) space in both the insertion-only and insertion-deletion settings.

1.4 Outline

We start with notations and definitions in Section 2. This section also introduces the necessary context on communication complexity needed in this work. In Section 3, we give our insertion-only streams algorithm, and in Section 4, we present a matching lower bound. Our algorithm for insertion-deletion streams is given in Section 5, and we conclude with a matching lower bound in Section 6.

2 PRELIMINARIES

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 *neighbourhood* 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 FEwW is to output a neighbourhood of size at least d/α.

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 overview on information theory we refer the reader to [18].

Communication Complexity

We now provide the necessary context on communication complexity (see [32] for more information).

In the *one-way p-party communication model*, for $p \ge 2$, p parties $P_1, P_2, \dots, P_p$ 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 $(X_1, X_2, \dots, X_p)$ 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, $\max_{1 \le i \le p-1}\{|M_i|\}$, where $|M_i|$ is the maximum length of message M_i. The *randomized one-way communication complexity* $R_{\epsilon}^{\rightarrow}(\text{Prob})$ 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 $D_{\mathcal{D},\epsilon}^{\rightarrow}(\text{Prob})$, 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 input distribution $\mathcal{D}$. In order to prove lower bounds on $R_{\epsilon}^{\rightarrow}(\text{Prob})$, by Yao's lemma it is enough to bound the distributional communication complexity for any suitable input distribution since $R_{\epsilon}^{\rightarrow}(\text{Prob}) = \max_{\mathcal{D}} D_{\mathcal{D},\epsilon}^{\rightarrow}(\text{Prob})$. 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. [5, 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, since 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 $j \ge i + 1$. This then implies a lower bound on the communication complexity of a specific problem Prob since $I_{\mathcal{D}}(X_i : M_j) \le H_{\mathcal{D}}(M_j) \le |M_j|$ holds for any protocol.

3 ALGORITHM FOR INSERTION-ONLY STREAMS

Before presenting our algorithm for FEwW 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 neighbourhood of size $\min\{d_2, \deg - d_1 + 1\}$, where deg is the degree of the respective node. If at least one neighbourhood of size d_2 is found then we say that the algorithm *succeeds* and returns an arbitrary neighbourhood among the stored neighbourhoods 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 fulfils 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 d_2 such edges are found.

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 $\ge d_1$}
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 prob. $\frac{s}{x}$}
        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 edge}
    $S \leftarrow S \cup \{ab\}$
return Arbitrary neighbourhood among those of size d_2 in S, if there is none return `fail`

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 neighbours 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*

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

Proof. Let $D \subseteq V$ be the set of vertices of degree at least d_1 (then $|D| \le n_1$). First, suppose that $d_1 \le s$. 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). Hence, a neighbourhood of size d_2 is necessarily found.

Otherwise, by well-known properties of reservoir sampling (e.g. [38]), 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 degree at least $d_1 + d_2 - 1$ is sampled is at most:

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

□

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 α-approximation Streaming Algorithm for FEwW

Require: Degree bound d, approximation factor α

$s \leftarrow \lceil \ln(n) \cdot n^{\frac{1}{\alpha}} \rceil$
for $i = 0 \ldots \alpha - 1$ **do in parallel**
  $(a_i, S_i) \leftarrow$ Deg-Res-Sampling$(\max\{1, i \cdot \frac{d}{\alpha}\}, \frac{d}{\alpha}, s)$
return Any neighbourhood (a_i, S_i) among the successful runs

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 $\alpha \ge 2$, Algorithm 2 finds a neighbourhood of size $\frac{d}{\alpha}$ with probability at least $1 - \frac{1}{n}$ and uses space*

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

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 $m = \mathrm{poly}\, n$). The algorithm runs the subroutine Deg-Res-Sampling (Algorithm 1) α times in parallel. Each of these runs requires space $\mathrm{O}(s \cdot \frac{d}{\alpha} \log n)$. Besides the vertex degrees, we thus have an additional space requirement of

$O(s \cdot d \log n) = O(n^{\frac{1}{\alpha}} d \log^2 n)$ 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 $i \cdot \frac{d}{\alpha}$. Observe that $n \geq n_0 \geq n_1 \geq n_2 \geq \cdots \geq n_\alpha \geq 1$, 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 $0 \leq i \leq \alpha - 1$:

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

Since $n_0 \leq n$ we obtain:

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

and since $n_c \geq 1$ we have:

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

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

3.3 Extension to Star Detection

Streaming algorithms for FEwW can be used to solve Star Detection with a small increase in space and approximation ratio.

Lemma 3.3. *Let* **A** *be a one-pass α-approximation streaming algorithm for FEwW with space $s(n, d)$ that succeeds with probability $1 - \delta$. Then there exists a one-pass $(1 + \epsilon)\alpha$-approximation streaming algorithm for Star Detection with space $O(s(n, n) \cdot \log_{1+\epsilon} n)$ that succeeds with probability $1 - \delta$.*

Proof. Let $G = (V, E)$ be the graph described by the input stream in an instance of Star Detection. We use $O(\log_{1+\epsilon} n)$ 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 algorithm **A** for FEwW 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 finds a neighbourhood of size at least $\Delta'/\alpha \geq \Delta/(\alpha(1 + \epsilon))$ and thus a large star in G. □

The previous result in combination with Theorem 3.2 can be used to obtain a semi-streaming algorithm for Star Detection (by using any fixed constant ϵ and $\alpha = \log n$ in the previous lemma).

Corollary 3.4. *There is a semi-streaming $O(\log n)$-approximation algorithm for Star Detection that succeeds with high probability.*

4 LOWER BOUND FOR INSERTION-ONLY STREAMS

In this section, we first point out that a simple $\Omega(n/\alpha^2)$ 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 FEwW yields a protocol for Bit-Vector Learning, which gives the desired lower bound.

4.1 An $\Omega(n/\alpha^2)$ 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:

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

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

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

Proof. Let $(S_1, S_2, \ldots, S_p)$ be an instance of Set-Disjointness$_p$. For $\alpha = p/1.01$, let **A** be an α-approximation streaming algorithm for FEwW, and let $d = k \cdot p$, for some integer $k \geq 1$. The parties use **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 **A** on their edges E_1, sends the resulting memory state to party 2 who continues running **A** on E_2. This continues until party p completes the algorithm. Since **A** is a $p/1.01$-approximation algorithm, if the sets uniquely intersect, the output of the algorithm is a neighbourhood of size at least $\lceil \frac{\Delta}{\alpha} \rceil = \lceil 1.01 \cdot k \rceil \geq k+1$. If the sets are disjoint, then no neighbourhood 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 $\Omega(n/p^2)$, **A** uses space $\Omega(n/p^2) = \Omega(n/\alpha^2)$.

□

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:*[3] 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 [3] 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. [4]

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 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 ease of presentation, we will only consider values of 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 [7], which is stated as Theorem 4.4 below and requires this property. This can be avoided using additional case distinctions.

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

[4]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.

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 (i.e., at least** 6 **positions) of one of the strings** $Z^1 = 1001011011$, $Z^2 = 01000$, $Z^3 = 01011$, **or** $Z^4 = 011110101000011$.

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 neighbourhood 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:*

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

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:

$$\begin{aligned} H(E, Z^I \mid Out) &= H(Z^I \mid Out) + H(E \mid Out, Z^I) \\ &= H(Z^I \mid Out) , \end{aligned} \tag{1}$$

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

$$\begin{aligned} H(E, Z^I \mid Out) &= H(E \mid Out) + H(Z^I \mid E, Out) \\ &\leq 1 + H(Z^I \mid E, Out) , \end{aligned} \tag{2}$$

using the chain rule for entropy and the bound $H(E \mid Out) \leq H(E) \leq 1$ (conditioning reduces entropy). From Inequalities 1 and 2 we obtain:

$$H(Z^I \mid Out) \leq 1 + H(Z^I \mid E, Out) . \tag{3}$$

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

$$\begin{aligned} H(Z^I \mid E, Out) &= \mathbb{P}[E = 0] H(Z^I \mid Out, E = 0) \\ &\quad + \mathbb{P}[E = 1] H(Z^I \mid Out, E = 1) . \end{aligned} \tag{4}$$

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:

$$\begin{aligned} H(Z^I \mid E, Out) &\leq (1-\epsilon)(H(Z^I) - 1.01k) + \epsilon H(Z^I) \\ &= H(Z^I) - (1-\epsilon)1.01k \, . \end{aligned} \tag{5}$$

Next, using Inequalities 3 and 5, 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 [7], which in its original form states that every complete regular hypergraph is 1-factorisable, 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 [7] - 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 $S_1, S_2, \dots, S_{|S|\frac{k}{n}}$ such that:*

(1) $|S_i| = \frac{n}{k}$, for every i,
(2) $S_i \cap S_j = \emptyset$, 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:*

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

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:

$$\begin{aligned} & I(M_{i-1} : Y_{i-1}^{X_i} \mid X_i X_{i-1}) \\ &= \mathop{\mathbb{E}}_{x_{i-1} \leftarrow X_{i-1}} \mathop{\mathbb{E}}_{x_i \leftarrow X_i} I(M_{i-1} \ : \ Y_{i-1}^{X_i} \mid X_i = x_i, X_{i-1} = x_{i-1}) \\ &= \mathop{\mathbb{E}}_{x_{i-1} \leftarrow X_{i-1}} \mathop{\mathbb{E}}_{x_i \leftarrow X_i} I(M_{i-1} \ : \ Y_{i-1}^{x_i} \mid X_{i-1} = x_{i-1}) \, . \end{aligned} \tag{6}$$

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 [7] as stated in 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 $\mathcal{X}_1(x_{i-1}), \mathcal{X}_2(x_{i-1}), \dots$ such that $\cup_{x \in \mathcal{X}_j(x_{i-1})} x = x_{i-1}$. Denote the elements of set $\mathcal{X}_j(x_{i-1})$ by $x_j^1, x_j^2, \dots, x_j^{\frac{n_{i-1}}{n_i}}$. We thus have:

$$\begin{aligned} & \mathop{\mathbb{E}}_{x_i \leftarrow X_i} I(M_{i-1} \ : \ Y_{i-1}^{x_i} \mid X_{i-1} = x_{i-1}) \\ &= \frac{1}{|\mathcal{X}(x_{i-1})|} \sum_{x_i \in \mathcal{X}(x_{i-1})} I(M_{i-1} \ : \ Y_{i-1}^{x_i} \mid X_{i-1} = x_{i-1}) \\ &= \frac{1}{|\mathcal{X}(x_{i-1})|} \sum_{j \in [|\mathcal{X}(x_{i-1})| \frac{n_i}{n_{i-1}}]} \sum_{\ell \in [\frac{n_{i-1}}{n_i}]} I(M_{i-1} \ : \ Y_{i-1}^{x_j^\ell} \mid X_{i-1} = x_{i-1}) \\ &\leq \frac{1}{|\mathcal{X}(x_{i-1})|} \sum_{j \in [|\mathcal{X}(x_{i-1})| \frac{n_i}{n_{i-1}}]} \sum_{\ell \in [\frac{n_{i-1}}{n_i}]} I(M_{i-1} \ : \ Y_{i-1}^{x_j^\ell} \mid Y_{i-1}^{x_j^1} \dots \\ & \qquad \dots Y_{i-1}^{x_j^{\ell-1}}, X_{i-1} = x_{i-1}) \\ &= \frac{1}{|\mathcal{X}(x_{i-1})|} \sum_{j \in [|\mathcal{X}(x_{i-1})| \frac{n_i}{n_{i-1}}]} I(M_{i-1} \ : \ Y_{i-1} \mid X_{i-1} = x_{i-1}) \\ &= \frac{n_i}{n_{i-1}} I(M_{i-1} \ : \ Y_{i-1} \mid X_{i-1} = x_{i-1}) \, , \end{aligned} \tag{7}$$

where we used Lemma 4.2 to obtain the first inequality and the chain rule for mutual information for the subsequent equality. Combining Inequalities 6 and 7, 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 $Y_1^{X_2}, Y_2^{X_3}, \dots, Y_{p-1}^{X_p}$ is limited.

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

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

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

$$\begin{aligned} & I(M_{i-1} X_i Y_i \ : \ Y_1^{X_2} \dots Y_{i-1}^{X_i}) \\ &= I(X_i Y_i \ : \ Y_1^{X_2} \dots Y_{i-1}^{X_i}) + I(M_{i-1} \ : \ Y_1^{X_2} \dots Y_{i-1}^{X_i} \mid X_i Y_i) \\ &= 0 + I(M_{i-1} \ : \ Y_1^{X_2} \dots Y_{i-1}^{X_i} \mid X_i) \, , \end{aligned} \tag{8}$$

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:

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

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

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

We have thus shown:

$$\begin{aligned} I(M_{i-1} \ : \ Y_1^{X_2} \dots Y_{i-1}^{X_i} \mid X_i) \\ \leq |M_{i-1}| \frac{n_i}{n_{i-1}} + I(M_{i-2} \ : \ Y_1^{X_2} \dots Y_{i-2}^{X_{i-1}} \mid X_{i-1}) \ . \end{aligned} \tag{9}$$

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

$$I(M_1 \ : \ Y_1^{X_2} \mid X_2) \leq |M_1| \frac{n_2}{n_1} \ . \tag{10}$$

Using Equality 8 and Inequalities 9 and 10, we obtain:

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

□

Finally we are ready to prove 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:*

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

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

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

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: FEwW to Bit-Vector Learning

In this subsection, we show that a streaming algorithm for FEwW can be used to obtain 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 FEwW.

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

$$\Omega(\frac{d n^{\frac{1}{p-1}}}{\alpha^2}) \ .$$

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:

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

An illustration of this construction is given in Figure 2 (the example uses the notation $B_i = \{2k(p-1)+1, \dots, 2kp\}$). Observe that $\Delta = kp$ (the vertex in X_p has such a degree).

Let $\mathbf{A}$ be an α-approximation streaming algorithm for FEwW(n, d) with $\alpha = \frac{p}{1.01}$ 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 the algorithm and outputs a neighbourhood (I, S). We observe that every neighbour $s \in S$ 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

$$\Omega(\frac{k n^{\frac{1}{p-1}}}{p}) = \Omega(\frac{d n^{\frac{1}{p-1}}}{\alpha^2}) \ ,$$

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 FEwW 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 $1 - \delta$ [26].

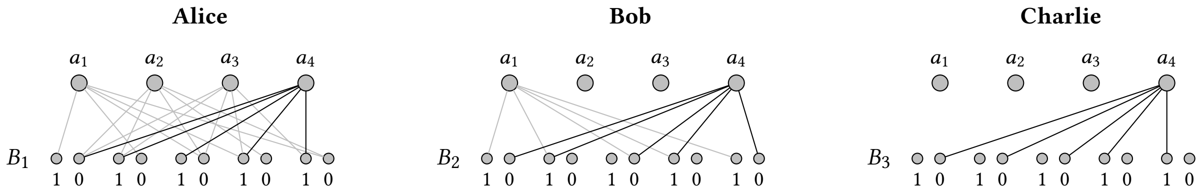

Figure 2: In the example instance given in Figure 1, Alice holds $Y_1^1 = 10010$, $Y_1^2 = 01000$, $Y_1^3 = 01011$, and $Y_1^4 = 01111$. For each string Y_1^j, Alice connects vertex a_j to 5 vertices, each indicating one bit of the respective bit string. For example, when reading the labels of the B_1-vertices connected to a_4 from left-to-right, we obtain the bit sequence 01111 which equals Y_1^4.

In the following, we will run $\tilde{O}(nd)$ 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 $x = \max\{\frac{n}{\alpha}, \sqrt{n}\}$
(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 $10\frac{d}{\alpha} \ln n$ l_0-samplers on the set of edges incident to a. This strategy requires space $\tilde{O}(\frac{xd}{\alpha})$.
(3) **Edge Sampling:** Run $10\frac{nd}{\alpha}\left(\frac{1}{x} + \frac{1}{\alpha}\right)\ln(nm)$ l_0-samplers on the stream, each producing a uniform random edge. This strategy requires space $\tilde{O}\left(\frac{nd}{\alpha}\left(\frac{1}{x} + \frac{1}{\alpha}\right)\right)$.
(4) Output any neighbourhood of size at least $\frac{d}{\alpha}$ 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 $y \le k \le n$. 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 $C \ge 4$. Then, $|Y \cap X| \ge y$ with probability $1 - \frac{1}{n^{C-3}}$.*

We will first show that if the input graph contains enough vertices of degree at least $\frac{d}{\alpha}$, 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 $\frac{d}{\alpha}$.*

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

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

Next, suppose that there is a node $a \in A'$ with $\deg(a) \ge \frac{d}{\alpha}$. Then, by Lemma 5.1 sampling $10 \cdot \frac{d}{\alpha} \log n$ times uniformly at random from the set of edges incident to a results in at least $\frac{d}{\alpha}$ 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 $\frac{d}{\alpha}$.*

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 $\frac{d}{\alpha}$, the input graph has at most $|E| \le \frac{n}{x} \cdot \Delta + n \cdot \frac{d}{\alpha}$ edges. Fix now a node a of degree Δ. Then, by Lemma 5.1, we will sample $\frac{d}{\alpha}$ different edges incident to a with high probability, if we sample

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

times, 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 α-approximation streaming for insertion-deletion streams that uses space $\tilde{O}(\frac{dn}{\alpha^2})$ if $\alpha \le \sqrt{n}$, and space $\tilde{O}(\frac{\sqrt{n}d}{\alpha})$ if $\alpha > \sqrt{n}$, 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 $\tilde{O}(\frac{xd}{\alpha}) + \tilde{O}(\frac{nd}{\alpha}(\frac{1}{x} + \frac{1}{\alpha}))$, which simplifies to the bounds claimed in the statement of the theorem by choosing $x = \max\{\frac{n}{\alpha}, \sqrt{n}\}$. □

Using the same ideas as in the proof of Corollary 3.4, we obtain:

Corollary 5.5. *There is a* $O(\sqrt{n})$*-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 FEwW in insertion-deletion streams. To this end, we first define the two-party communication problem Augmented-Matrix-Row-Index and prove a lower bound on its communication complexity. Finally, we argue that an insertion-deletion streaming algorithm for FEwW 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 by 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 the problem 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 $Y_J = \perp$ 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.

$$\textbf{Alice} \qquad \xrightarrow{M} \qquad \textbf{Bob}$$

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 1 & & & 0 \\ 1 & 1 & & 0 & 1 & \\ & & & & & \\ & 0 & 1 & 0 & & 0 \end{pmatrix}$$

Figure 3: Example Instance of 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.

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:*

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

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:

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

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 $Z_J = \varnothing$. Furthermore, let L be a random variable that is uniformly distributed in $[n] \setminus J$. Consider now a fixed index j. Then, using the chain rule for mutual information and the fact that X_{Z_i} and X_{Z_q} are independent, for every $i \neq q$, we obtain:

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

By combining all potential values for j, we obtain:

$$I(X_Z : M \mid X_J X_Y) \geq (n-1) \cdot I(X_{Z_L} : M \mid X_J X_Y L) .$$

In the following, we will show that $I(X_{Z_L} : M \mid X_J X_Y L) \geq k - 1 - \epsilon m$, 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

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

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:

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

Last, it remains to argue that $I(X_{Z_L} : M \mid X_{Y'_J} X_{Y \setminus Y_L} L)$ is equivalent to $I(X_{Z_J} : M \mid JYX_Y)$. 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:

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

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: FEwW to Augmented-Matrix-Row-Index

LEMMA 6.3. *Let* $\mathbf{A}$ *be an* α*-approximation insertion-deletion streaming algorithm for FEwW*(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*$(n, 2d, \frac{d}{\alpha} - 1)$ *with message size*

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

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

PROOF. We will show how algorithm $\mathbf{A}$ can be used to solve Augmented-Matrix-Row-Index$(n, 2d, \frac{d}{\alpha} - 1)$. 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 $\Theta(\alpha \log n)$ times in parallel:

First, Alice and Bob use public randomness to chose n permutations $\pi_i : [2d] \rightarrow [2d]$ 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 FEwW(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 neighbourhood of size at least $\frac{d}{\alpha}$. Observe that after Bob's deletions, every row except row J contains at most $\frac{d}{\alpha} - 1$ 1s, which implies that $\mathbf{A}$ reports a neighbourhood rooted at A-vertex J (the vertex that corresponds to row J). Bob thus learns at least $\frac{d}{\alpha}$ positions of row J where the matrix X' is 1. Bob then applies $(\pi_J)^{-1}$ and thus learns at least $\frac{d}{\alpha}$ 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 $\frac{d/\alpha}{2d} = \frac{1}{2\alpha}$. Applying concentration bounds, since the protocol is repeated $\Theta(\alpha \cdot \log n)$ 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$(n, 2d, \frac{d}{\alpha} - 1)$.

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. □

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

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$(n, 2d, \frac{d}{\alpha} - 1)$ that succeeds with probability $\delta + n^{-10}$ and communicates $O(s \cdot \alpha \log n)$ bits. Then, by Theorem 6.2, we have:

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

which implies

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

□

REFERENCES

[1] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. 2012. Analyzing Graph Structure via Linear Measurements. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms* (Kyoto, Japan) *(SODA '12)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 459–467. http://dl.acm.org/citation.cfm?id=2095116.2095156

[2] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. 2019. Sublinear Algorithms for $(\Delta + 1)$ Vertex Coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (San Diego, California) *(SODA '19)*. Society for Industrial and Applied Mathematics, USA, 767–786.

[3] Sepehr Assadi, Sanjeev Khanna, and Yang Li. 2016. 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*. 698–711. https://doi.org/10.1145/2897518.2897576

[4] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. 2016. 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*. 1345–1364. https://doi.org/10.1137/1.9781611974331.ch93

[5] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. 2002. An Information Statistics Approach to Data Stream and Communication Complexity. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS '02)*. IEEE Computer Society, Washington, DC, USA, 209–218. http://dl.acm.org/citation.cfm?id=645413.652164

[6] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. 2002. Information theory methods in communication complexity. In *Proceedings 17th IEEE Annual Conference on Computational Complexity*. 93–102. https://doi.org/10.1109/CCC.2002.1004344

[7] Zsolt Baranyai. 1979. The edge-coloring of complete hypergraphs I. *Journal of Combinatorial Theory, Series B* 26, 3 (1979), 276 – 294. https://doi.org/10.1016/0095-8956(79)90002-9

[8] Radu Berinde, Graham Cormode, Piotr Indyk, and Martin J. Strauss. 2009. Space-optimal Heavy Hitters with Strong Error Bounds. In *Proceedings of the Twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Providence, Rhode Island, USA) *(PODS '09)*. ACM, New York, NY, USA, 157–166. https://doi.org/10.1145/1559795.1559819

[9] Radu Berinde, Piotr Indyk, Graham Cormode, and Martin J. Strauss. 2010. Space-optimal heavy hitters with strong error bounds. *ACM Trans. Database Syst.* 35, 4 (2010), 26:1–26:28. https://doi.org/10.1145/1862919.1862923

[10] Arnab Bhattacharyya, Palash Dey, and David P. Woodruff. 2018. An Optimal Algorithm for &Ell;1-Heavy Hitters in Insertion Streams and Related Problems. *ACM Trans. Algorithms* 15, 1, Article 2 (Oct. 2018), 27 pages. https://doi.org/10.1145/3264427

[11] Yousra Chabchoub, Christine Fricker, and Hanene Mohamed. 2009. Analysis of a Bloom Filter algorithm via the supermarket model. In *21st International Teletraffic Congress, ITC 2009, Paris, France, September 15-17, 2009*. 1–8. http://ieeexplore.ieee.org/document/5300252/

[12] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. 2003. 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*. 107–117. https://doi.org/10.1109/CCC.2003.1214414

[13] Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew Chi-Chih Yao. 2001. 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. 270–278. https://doi.org/10.1109/SFCS.2001.959901

[14] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. Finding Frequent Items in Data Streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP '02)*. Springer-Verlag, Berlin, Heidelberg, 693–703. http://dl.acm.org/citation.cfm?id=646255.684566

[15] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. 2004. Finding frequent items in data streams. *Theor. Comput. Sci.* 312, 1 (2004), 3–15. https://doi.org/10.1016/S0304-3975(03)00400-6

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

[17] Graham Cormode and S. Muthukrishnan. 2005. An Improved Data Stream Summary: The Count-min Sketch and Its Applications. *J. Algorithms* 55, 1 (April 2005), 58–75. https://doi.org/10.1016/j.jalgor.2003.12.001

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

[19] Jacques Dark and Christian Konrad. 2020. Optimal Lower Bounds for Matching and Vertex Cover in Dynamic Graph Streams. In *35th Computational Complexity Conference, CCC 2020 (LIPIcs)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[20] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. 2002. Frequency Estimation of Internet Packet Streams with Limited Space. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA '02)*. Springer-Verlag, Berlin, Heidelberg, 348–360.

[21] Cristian Estan and George Varghese. 2003. New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice. *ACM Trans. Comput. Syst.* 21, 3 (Aug. 2003), 270–313. https://doi.org/10.1145/859716.859719

[22] Shir Landau Feibish, Yehuda Afek, Anat Bremler-Barr, Edith Cohen, and Michal Shagam. 2017. 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*. 8:1–8:6. https://doi.org/10.1145/3132465.3132474

[23] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. 2005. On graph problems in a semi-streaming model. *Theoretical Computer Science* 348, 2 (2005), 207 – 216. https://doi.org/10.1016/j.tcs.2005.09.013 Automata, Languages and Programming: Algorithms and Complexity (ICALP-A 2004).

[24] Magnús M. Halldórsson, Xiaoming Sun, Mario Szegedy, and Chengu Wang. 2012. Streaming and Communication Complexity of Clique Approximation. In *Automata, Languages, and Programming*, Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 449–460.

[25] Monika R. Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. 1999. External Memory Algorithms. American Mathematical Society, Boston, MA, USA, Chapter Computing on Data Streams, 107–118. http://dl.acm.org/citation.cfm?id=327766.327782

[26] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. 2011. 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* (Athens, Greece) *(PODS '11)*. ACM, New York, NY, USA, 49–58. https://doi.org/10.1145/1989284.1989289

[27] Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Navid Nouri, Aaron Sidford, and Jakab Tardos. 2020. Fast and Space Efficient Spectral Sparsification in Dynamic Streams. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms* (Salt Lake City, Utah) *(SODA '20)*. Society for Industrial and Applied Mathematics, USA, 1814–1833.

[28] Richard M. Karp, Scott Shenker, and Christos H. Papadimitriou. 2003. A Simple Algorithm for Finding Frequent Elements in Streams and Bags. *ACM Trans. Database Syst.* 28, 1 (2003), 51–55. https://doi.org/10.1145/762471.762473

[29] Christian Konrad. 2015. Maximum Matching in Turnstile Streams. In *Algorithms - ESA 2015*, Nikhil Bansal and Irene Finocchi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 840–852.

[30] Christian Konrad, Frédéric Magniez, and Claire Mathieu. 2012. Maximum Matching in Semi-streaming with Few Passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Anupam Gupta, Klaus Jansen, José Rolim, and Rocco Servedio (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 231–242.

[31] Abhishek Kumar and Jun (Jim) Xu. 2006. 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*. https://doi.org/10.1109/INFOCOM.2006.326

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

[33] Gurmeet Singh Manku and Rajeev Motwani. 2002. Approximate Frequency Counts over Data Streams. In *Proceedings of the 28th International Conference on Very Large Data Bases* (Hong Kong, China) *(VLDB '02)*. VLDB Endowment, 346–357. http://dl.acm.org/citation.cfm?id=1287369.1287400

[34] Andrew McGregor. 2014. Graph Stream Algorithms: A Survey. *SIGMOD Rec.* 43, 1 (May 2014), 9–20. https://doi.org/10.1145/2627692.2627694

[35] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. 2005. Efficient Computation of Frequent and Top-k Elements in Data Streams. In *Database Theory - ICDT 2005, 10th International Conference, Edinburgh, UK, January 5-7, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3363)*, Thomas Eiter and Leonid Libkin (Eds.). Springer, 398–412. https://doi.org/10.1007/978-3-540-30570-5_27

[36] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. 2005. Efficient Computation of Frequent and Top-k Elements in Data Streams. In *Proceedings of the 10th International Conference on Database Theory* (Edinburgh, UK) *(ICDT'05)*. Springer-Verlag, Berlin, Heidelberg, 398–412. https://doi.org/10.1007/978-3-540-30570-5_27

[37] Jayadev Misra and David Gries. 1982. Finding Repeated Elements. *Sci. Comput. Program.* 2, 2 (1982), 143–152. https://doi.org/10.1016/0167-6423(82)90012-0

[38] Jeffrey S. Vitter. 1985. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.* 11, 1 (March 1985), 37–57. https://doi.org/10.1145/3147.3165

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:

$$H(E, X_J \mid Out) = H(X_J \mid Out) + H(E \mid Out, X_J) = H(X_J \mid Out) , \quad (11)$$

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

$$\begin{aligned} H(E, X_J \mid Out) &= H(E \mid Out) + H(X_J \mid E, Out) \\ &\leq 1 + H(X_J \mid E, Out) , \end{aligned} \quad (12)$$

using the chain rule for entropy and the bound $H(E \mid Out) \leq H(E) \leq 1$. From Inequalities 11 and 12 we obtain:

$$H(X_J \mid Out) \quad \leq \quad 1 + H(X_J \mid E, Out) . \quad (13)$$

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

$$\begin{aligned} H(X_J \mid E, Out) = \mathbb{P}[E = 0]\, H(X_J \mid Out, E = 0) \\ + \mathbb{P}[E = 1]\, H(X_J \mid Out, E = 1) . \end{aligned} \quad (14)$$

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:

$$H(X_J \mid E, Out) \quad \leq \quad (1 - \epsilon) \cdot 0 + \epsilon H(X_J) = \epsilon H(X_J) . \quad (15)$$

Next, using Inequalities 13 and 15, 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. □