Counting distinct items over update streams

Sumit Ganguly

Indian Institute of Technology, Kanpur

(Extended Abstract)

Abstract. In data streaming applications, data arrives at rapid rates and in high volume, thus making it essential to process each stream update very efficiently in terms of both time and space. We consider the problem of tracking the number of distinct items (also denoted as F_0) over *update* streams, that is, streams with general insertion and deletion operations. We present two algorithms that improve on the space and time complexity of existing algorithms for estimating F_0 for update streams [1, 2, 8, 9].

1 Introduction

Data streaming applications occur naturally in many established and emerging application areas, such as database systems, network monitoring, sensor networks etc.. These applications are characterized by rapidly arriving voluminous data and require algorithms that, (a) have low time complexity of processing each stream update, and, (b) have sub-linear space complexity. In this paper, we consider a class of data streaming applications that correspond to the *update* model of streaming, that is, records arriving over a stream correspond to both insertion and deletion of data (e.g., establishment and termination of a network connection). In particular, consider the problem of estimating the number of distinct items that have arrived over an update stream. Queries that count the number of distinct items in a stream(s) satisfying given predicates play a significant role in decision-support applications. For example, consider the following network monitoring query: find the number of distinct connections that have been routed through router R_1 and R_2 but not through router R_3.

We view *update data streams* as a sequence of arrivals of the form (i, v), where, i is the identity of an item, assumed to be from the domain $\mathcal{D} = \{0, 1, \ldots, N-1\}$, and v is the change in frequency of i. That is, $v > 0$ specifies v insertions of the item i and $v < 0$ correspondingly specifies v deletions of i. The frequency of an item i, denoted by f_i, is defined as $f_i = \sum_{(i,v)\in\mathcal{S}} v$. We assume that items cannot have negative frequency. The metric F_0 of a stream is defined as the number of distinct items i in the stream whose frequency f_i is positive (i.e., the zeroth frequency moment $F_0 = \sum_{f_i>0} f_i^0$). A canonical problem in the class of distinct item queries is to estimate F_0. The problem of estimating the size of distinct item queries over stream(s) is a straightforward generalization of the problem of estimating F_0 [8]. We therefore restrict ourselves to the problem of estimating F_0.

Previous work. The append only model of streaming data refers to streams without deletion operations. . The problem of estimating F_0 has received considerable attention for append-only data streams [7, 1, 2, 4, 5, 10–12, 3, 13]. The algorithm of [3] is the most time and space efficient among the algorithms that are applicable to append-only streams only. [13] presents a lower bound of $\Omega(\frac{1}{\epsilon^2})$ for estimating F_0 to within an accuracy factor of $1 \pm \epsilon$. (We use $\epsilon > 0$ and $0 < \delta < 1$ to denote the accuracy and confidence parameters, respectively) The algorithms of [10–12, 3] use the technique of distinct samples[10–12, 3], which is a simple and elegant technique for estimating F_0.

Algorithms for estimating F_0 over streams with deletion operations [1, 2, 8, 9] use the technique of hashing introduced in [7]. The algorithms of [1, 2, 8] use space $O(\frac{1}{\epsilon^2} \cdot \log N \log m \cdot \log \frac{1}{\delta})$ bits (where, m is the sum of frequencies of the items in stream) and are the most space-economical among the algorithms for estimating F_0 over update streams. However, they require time $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ operations for processing each stream update. The algorithm of [9] is the most time-efficient among algorithms that estimate F_0 over update streams and requires time $O(\log N \cdot (\log \frac{1}{\epsilon} + \log \frac{1}{\delta}))$ to process each stream update. However, it uses an additional factor of $O(\log N \log \frac{1}{\delta})$ in space, as compared to [1, 2, 8], that is, its space complexity is $O(\frac{1}{\epsilon^2} \log^2 N \log m \log^2 \frac{1}{\delta})$ bits. This algorithm designs an updatable distinct sample, which is a data structure that can be used for both insertions and deletions of items and effectively yields a distinct sample [10–12, 3].

Our Contributions. We present two novel algorithms for estimating F_0 over update streams that improves upon the time and space complexity of existing algorithms. Our first algorithm has time complexity $O(\log \frac{1}{\epsilon} + \log \frac{1}{\delta})$, which is a logarithmic factor smaller than the time complexity of the most time efficient algorithm currently known for this problem [9]. Its space complexity is $O(\frac{1}{\epsilon^2}(\log m + \log N) \log N (\log \frac{1}{\delta} + \log \frac{1}{\epsilon}) \log \frac{1}{\delta})$ bits, which is lower than that of [9] by a logarithmic factor. The improvement is a consequence of a more efficient design of the updatable distinct samples structure. Our second algorithm for estimating F_0 presents a time versus space tradeoff vis-a-vis our first algorithm. It takes $O\left((\log \frac{1}{\epsilon})(\log \frac{1}{\delta})\right)$ time to process each stream update and requires space $O(\frac{1}{\epsilon^2}(\log N + \log m) \log N \log \frac{1}{\delta})$ bits, thus, using a logarithmic factor less space and an extra logarithmic factor in time, in comparison with our first algorithm.

Organization. Sections 2 and 3 present the first and second algorithms respectively, whereas Section 4 concludes the paper.

2 Algorithm I

The distinct samples data structure [10–12] was used to estimate F_0 over append-only or sliding window data streams. In this section, we generalize the structure so that it can be used to estimate F_0 for streams with deletion operations. We begin with the TestSingleton data structure.

2.1 TestSingleton data structure

A TestSingleton data structure supports two operations, namely, (a) an update operation corresponding to insertion and deletion operations over the stream, called TestSingletonUpdate, and, (b) a procedure TestCard that tests whether F_0 is either 0, 1 or greater than 1, and accordingly returns Empty, Singleton or Collision, respectively. Further, if TestCard returns Singleton, then the unique member i comprising the singleton set and its frequency f_i is also returned.

A straightforward counting argument shows that a TestSingleton structure must use $\Omega(\log N)$ bits. We keep three counters m, U and V (all initialized to zero) with the following properties.

$$m = \sum_{i \in \text{ stream}} f_i, \quad U = \sum_{i \in \text{ stream}} f_i \cdot i, \quad \text{and} \quad V = \sum_{i \in \text{stream}} f_i \cdot i^2 \ .$$

The counters are easily maintained in the face of stream updates as shown below.

```
TestSingletonUpdate($(i, v)$):  $m := m + v$; $U := U + v \cdot i$ ; $V := V + v \cdot i^2$;
```

The space required is $O(\log m + \log N)$ bits and the time required for the update operation is $O(1)$. The operation TestCard is given below.

```
TestCard: if  $(m = 0)$ return Empty
          else if  $(U^2 = m \cdot V)$ then return (Singleton, $\frac{U}{m}$, $m$)
          else  return Collision
```

The correctness of this test is proved below.

Theorem 1. *$F_0 = 1$ if and only if $m > 0$ and $U^2 = m \cdot V$. If $F_0 = 1$, then, the unique item in the stream has identity $\frac{U}{m}$ and has frequency m.*

Proof. Let the stream consist of a single item i with frequency $f_i > 0$. Then, $m = f_i > 0$, $U = m \cdot i$ and $V = m \cdot i^2$. Therefore, $U^2 = m \cdot V$, $m > 0$ and $i = \frac{U}{m}$.

To prove the converse, let x be a random variable such that for $0 \leq i \leq N-1$, $x = i$ with probability $\frac{f_i}{m}$. Then, $\mathrm{E}[x] = \sum_{i=0}^{N-1} \frac{f_i}{m} \cdot i$ and $\mathrm{E}[x^2] = \sum_{i=1}^{N-1} \frac{f_i}{m} \cdot i^2$. Since $m > 0$, the condition, $U^2 = mV$ is equivalent to the condition, $(\mathrm{E}[x])^2 = \mathrm{E}[x^2]$, or that, $\mathrm{Var}[x] = 0$. Therefore, the set of items in the stream is either empty or consists of a singleton item. Since, $m > 0$, the stream is not empty. Therefore, $F_0 = 1$. □

2.2 *K*-set structure

A k-set structure, for parameter k, is a dictionary data structure that supports insertion and deletion of items and an operation Retrset, that either returns the set of items S present in the dictionary or returns nil. Further, if the number of items $|S|$ in the dictionary is at most k, then Retrset returns S (i.e., does not return nil in this case).

Classic dictionary structures including heaps, binary search trees, red-black trees, AVL trees etc., satisfy the requirements of a k-set data structure for all values of k. However, these structures require space $\Omega(|S|)$, whereas, we require a design whose space complexity is close to the lower bound of $\Omega(k \log \frac{N}{k})$ bits. A randomized k-set data structure takes a confidence parameter δ such that if $|S| \leq k$, then RETRSET returns S with probability $1-\delta$.

We now present a design for a k-set structure. Our structure is a two dimensional array $H[R \times B]$ and can be thought of as consisting of $R = \lceil \log \frac{k}{\delta} \rceil$ hash tables, each consisting of B buckets indexed from 0 through $B-1$, where, $B = 2k$. Each bucket $H[r,b]$ is a TESTSINGLETON data structure. The r^{th} hash table uses a randomly chosen pair-wise independent hash function $h_r : \{0,1,\ldots,N-1\} \rightarrow \{0,1,\ldots,B-1\}$; the random bits used by each of the R hash tables are independent. Further, we maintain a variable m that tracks the total sum of the frequencies, that is, $m = \sum_{i \in S} f_i$. The structure is initialized so that all counters are initially zeros. For every stream update of the form (i,v), and for every hash table index r, we hash the item to its bucket number $h_r(i)$ and propagate the update to the corresponding TESTSINGLETON structure in that bucket.

KSETUPDATE(i,v): $H[r, h_r(i)]$.TESTSINGLETONUPDATE(i,v), for $1 \leq r \leq R$.

The space used by the k-set structure is $O(k(\log m + \log N)\log \frac{k}{\delta})$ bits and the time complexity of processing each stream update is $O(\log \frac{k}{\delta})$.

The RETRSET operation works as follows. We iterate over each of the B buckets of each of the R hash tables checking whether the bucket $H[r,b]$ is a singleton or not, for $r = 1,\ldots,R$, $b = 0,\ldots,B-1$. If the bucket is a singleton, then the item and its frequency is retrieved using the TESTCARD subroutine of the TESTSINGLETON structure. In this manner, we retrieve the union $\hat{S}$ of all the distinct items and their frequencies. Next, we preform a verification step, that is, the frequencies of the (distinct) retrieved items are added and compared with m. If they match, then $\hat{S}$ is returned, otherwise, NIL is returned. The correctness of the RETRSET procedure is proved below.

Theorem 2. *If* RETRSET $\neq$ NIL*, then* RETRSET $= S$*. Further, if* $|S| \leq k$*, then* $\Pr\{\text{RETRSET} = S\} \geq 1-\delta$*.*

Proof. If a bucket $H[r,b]$ is singleton, then, the TESTSINGLETON structure correctly retrieves the item and its frequency. Therefore, the set $\hat{S}$ of retrieved items is always a subset of S and $\sum_{i \in \hat{S}} f_i \leq m$. RETRSET returns non-NIL only if $\sum_{i \in \hat{S}} f_i = m$, which implies that $\hat{S} = S$. Now suppose that $|S| = s \leq k$. Fix an item $i \in S$ and a hash table index r. Let C_i be the number of items from $S - \{i\}$ that collide with i. Then, $\mathrm{E}[C_i] = \sum_{j \in S, j \neq i} \Pr\{h_r(i) = h_r(j)\} = \frac{s-1}{B}$, by pair-wise independence of h_r. By Markov's inequality, $\Pr\{C_i = 0\} \geq 1 - \frac{s-1}{B} > \frac{1}{2}$, since, $s \leq k$ and $B \geq 2k$. Therefore, the probability that i is not returned from any of the R hash tables is less than $\frac{1}{2^R}$. Thus, the total error probability is at most $\frac{k}{2^R} \leq \delta$. □

2.3 Updatable distinct samples structure

A distinct sample of a stream with sampling probability p is a set D of items such that each of the F_0 distinct items in the stream has an equal and independent chance of p of being included in D. An *updatable distinct sample* is a data structure that enables the extraction of a distinct sample over streams with insertions and deletion operations, thereby enabling distinct sampling based estimation algorithms to be used for update streams.

Let $F = GF(2^d)$ be a field such that $|F| \geq N^2$, where, N is the size of the domain of the items in the stream. An updatable distinct sample, with *capacity parameter* s, can be implemented using an array $D[1, \ldots, \log|F|]$, where, each entry of the array, namely, $D[r]$, is a k-set structure with $k = s$. A d-wise independent random hash function $h : F \rightarrow F$ is chosen (d is a parameter) and is used to define the function level $: F \rightarrow \{1, \ldots, \log|F|\}$ function as follows. level$(i) = 1$ if $h(i) = 0$; otherwise, level$(i) = \text{lsb}(h(i))$. Here, lsb(a) denotes the least significant bit position of a. Corresponding to a stream update of the form (i, v), the data structure is updated by invoking the update operation of the s-set structure $D[\text{level}(i)]$.

DistSampUpdate(i, v): $D[\text{level}(i)]$.KsetUpdate(i, v)

The *current level* l_{curr} of the updatable distinct sample is the lowest value of $1 \leq l \leq \log N$ such that $D[l]$.Retrset is non-nil. The distinct sample is defined as $D[l_{\text{curr}}]$.Retrset. The space required to store an updatable distinct sample with capacity parameter s is $O(s \cdot (\log m + \log N) \cdot \log N \cdot \log \frac{s}{\delta})$ and the time required to process each stream update is $O(\log \frac{s}{\delta} + \log \frac{1}{\epsilon})$.[1]

F_0 can be estimated for append-only streams using distinct samples of size $s = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ items and using a degree of independence of $d = O(\log \frac{1}{\epsilon})$ for the hash family [3]. By using updatable distinct samples in place of distinct samples, we obtain an algorithm for estimating F_0 over update streams, that processes each stream update in time $O(\log \frac{1}{\epsilon} + \log \frac{1}{\delta})$ and uses space $O(\frac{1}{\epsilon^2} \cdot (\log m + \log N) \cdot \log N \cdot (\log \frac{1}{\delta} + \log \frac{1}{\epsilon}) \cdot \log \frac{1}{\delta})$ bits.

3 Algorithm II

In this section, we present our second algorithm[2] for estimating F_0.

The basic data structure is a two-dimensional table $T[L \times K]$, where, $L = \log\ |F|$ (F is a field containing $\{0, 1, \ldots, N-1\}$ and such that $|F| \geq N^2$) and $K = \frac{720}{\epsilon^2}$. We keep two random hash functions namely, $h : F \rightarrow F$ and $g : F \rightarrow \{0, 1, \ldots, K-1\}$ that are d-wise independent, where, $d = O(\log \frac{1}{\epsilon})$. h is used to define a randomized level function as in Section 2.3, that is, level$(i) = 1$ if $h(i) = 0$, else level$(i) = \text{lsb}(h(i))$. Corresponding to each stream update of the form (i, v), the data structure is updated as follows.

Algo2Update(i, v) : $T[\text{level}(i), g(i)]$.TestSingletonUpdate(i, v)

That is, the item i is first hashed using the hash function h to obtain its level, say l. Next, the item is hashed to a bucket index $0 \leq g(i) \leq K - 1$ and the

[1] In comparison, the updatable distinct samples structure of [9] requires space $O(s \cdot \log m \cdot \log^2 N \cdot \log \frac{s}{\delta})$ bits and time $O(\log N \cdot \log \frac{s}{\delta} + \log \frac{1}{\epsilon})$ to process each stream update.

[2] We have not attempted to optimize the constants used in the algorithm

TestSingleton structure at the entry $T[l, g(i)]$ is updated accordingly. We keep $R = 96 \cdot \log \frac{2}{\delta}$ independent copies of the basic data structure. Finally, we also keep a k-set structure, where, $k = \frac{126}{\epsilon^2}$, as explained in Section 2.2, and maintain it in the presence of stream updates. The space used by the data structure is $O(\frac{1}{\epsilon^2} \log N (\log m + \log N) \log \frac{1}{\delta})$ bits and the time complexity of processing each stream update is $O(\log \frac{1}{\epsilon} \cdot \log \frac{1}{\delta})$ operations.

Each copy of the basic data structure yields an estimate $\hat{F}_0$ for F_0 as follows. We find the lowest level l such that the number X of the singleton buckets at this level exceeds $\frac{12}{\epsilon^2}$ and the number Z of the non-empty buckets exceeds $\frac{7K}{8}$. If there is no such level, then, we return $\hat{F}_0 = \perp$. Otherwise, we (numerically) solve the equation $X = \frac{\hat{F}_0}{2^l}\left(1 - \frac{1}{K}\right)^{\frac{\hat{F}_0}{2^l} - 1}$ and return the smaller of the two possible roots of this equation as $\hat{F}_0$. The median of the observations, denoted $\hat{F}_0^{\text{median}}$ from the copies is computed (ignoring those copies which have returned $\perp$). If the median value is at least $\frac{126}{\epsilon^2}$, then, the median value is returned, otherwise, an estimate for F_0 is obtained from the parallel k-set structure and returned.

The procedure is different from past work [1–3, 8] in that, in previous work, the count of the number of singleton buckets (or, a similar measure, such as the number of non-empty buckets) was taken across *independent observations*. In our case, the sum is taken across the buckets in the same hash table, and hence, the observations are not even pair-wise independent. We are not aware of Chernoff-like tail inequalities that are applicable in such a scenario[3]. Theorem 3 states the correctness property of the algorithm. Its proof relies on Lemma 1, which is the main technical lemma of this section.

Theorem 3. *Suppose* $\epsilon \leq \frac{1}{8}$, $K = \frac{720}{\epsilon^2}$, $d = O(\log \frac{1}{\epsilon})$ *and* $R = 96 \log \frac{2}{\delta}$. *Then, the estimate returned by Algorithm II is within* $(1 \pm 3\epsilon)F_0$ *with probability at least* $1 - 2\delta$.

Proof. Suppose $\hat{F}_0^{\text{median}} \geq (1 + \frac{3}{4})\frac{72}{\epsilon^2} = \frac{126}{\epsilon^2}$. Then, by Lemma 1, this observation happens with probability at most δ, provided, $F_0 < \frac{72}{\epsilon^2}$. In this case, with probability $1 - \delta$, the algorithm returns the estimate from the k-set structure, where, $k = \frac{126}{\epsilon^2}$, which is exactly correct with probability $1 - \delta$. Otherwise, by Lemma 1, the algorithm returns $\hat{F}_0^{\text{median}}$, which is within $(1 \pm \epsilon)F_0$ with probability $1 - \delta$. □

Lemma 1. *Suppose* $\epsilon \leq \frac{1}{8}$, $K = \frac{720}{\epsilon^2}$, $F_0 \geq \frac{72}{\epsilon^2}$, $d = O(\log \frac{1}{\epsilon})$ *and* $R = 96 \cdot \log \frac{2}{\delta}$. *Then,* $\Pr\left\{|\hat{F}_0^{\text{median}} - F_0| \leq 2\epsilon F_0\right\} > 1 - \delta$.

Analysis. Fix a level l. Consider the set of items in the stream that hash to level l (i.e., level$(i) = l$). For each such item, let p denote the probability that an item i is placed in a given bucket, that is, $p = \frac{1}{K}$. Corresponding to bucket numbered i at level l, $1 \leq i \leq K$, we define an indicator variable x_i that is 1 if exactly one ball has mapped to this bucket and is 0 otherwise. Let X denote the number of

[3] [6] presents tail inequalities for sums of negatively dependent boolean variables that assume that the hash function that distributes the balls to the buckets is *fully independent*. The techniques of [6] do not apply to the scenario when hash functions have limited dependence.

singleton buckets at this level, that is, $X = \sum_{i=1}^{K} x_i$. If the random hash functions are chosen from a d-wise independent hash family, then, we denote the corresponding probability function by $\mathrm{Pr}_d\left\{\cdot\right\}$, and the corresponding expectations by $\mathrm{E}_d\left[\cdot\right]$. If the random hash function is chosen from a fully independent hash family, then, we denote the corresponding probability function by $\Pr\left\{\cdot\right\}$ and the corresponding expectations as $\mathrm{E}\left[\cdot\right]$. Suppose that the number of items that have hashed to level l is n, and let $\Pr\left\{x_i = 1\right\}$ be denoted by q, then, $q = \frac{n}{K}(1-\frac{1}{K})^{n-1}$, and $\mathrm{E}\left[X\right] = Kq$. The following lemma can be used to obtain bounds for $\mathrm{E}_d\left[X\right]$.

Lemma 2 ([3, 8]). *If h is $d = O(\log \frac{1}{\epsilon})$-wise independent and $q \le \frac{1}{4}$, then $|\mathrm{Pr}_d\left\{x_i = 1\right\} - q| \le \epsilon^4 q$. Therefore, $|\mathrm{E}_d\left[X\right] - Kq| \le \epsilon^4 Kq$.* □

The majority of the analysis is devoted to showing that $X \in (1 \pm \epsilon)\mathrm{E}\left[X\right]$ (with probability at least $\frac{7}{8}$). Lemma 3 justifies this line of argument by showing that if X is close to $\mathrm{E}\left[X\right]$, then, $\hat{F}_0(X)$, calculated as the smaller of the two roots of the equation $X = \frac{\hat{F}_0}{2^l}(1-\frac{1}{K})^{\frac{\hat{F}_0}{2^l}-1})$ is also close to F_0. We denote $f(x) = \frac{x}{2^l}(1-\frac{1}{K})^{\frac{x}{2^l}-1}$, so that, $f(F_0) = \mathrm{E}\left[X\right]$. Lemma 3 also shows that the gap between the two roots is substantial.

Lemma 3. *Let $x = F_0$ and $\epsilon \le \frac{1}{8}$. If $\frac{x}{2^l} \le \frac{5K}{24}$, $|f(x) - f(y)| \le \epsilon f(x)$ then either $|y - x| \le 2\epsilon x$ or $y > 12x$.*

Proof. Let $0 < \gamma \le 1$ and $x = F_0$. $f(x + \gamma x) = f(x)(1 + \gamma)\left(1 - \frac{1}{K}\right)^{x\gamma/2^l}$. Therefore, $|f(x + \gamma x) - f(x)| = f(x)|((1 + \gamma)\left(1 - \frac{1}{K}\right)^{x\gamma/2^l} - 1)| \ge f(x)((1 + \gamma)(1 - \frac{\gamma}{4}) - 1) \ge f(x)\frac{14\gamma}{24}$. Since, $|f(x + \gamma x) - f(x)| \le \epsilon f(x)$, therefore, $\epsilon f(x) \ge \frac{14}{24}|\gamma| f(x)$, or that, $\gamma \le \frac{24}{14}\epsilon$. If $-1 < \gamma < 0$, then a similar analysis holds.

Another solution to $f(x + \gamma x) = (1 + \epsilon)f(x)$ occurs when $\gamma \ge \gamma'$, where, γ' satisfies the equation $\frac{1+\gamma'}{1+\epsilon} = e^{\frac{5\gamma'}{24}}$. For $\epsilon \le \frac{1}{8}$, $\gamma' > 11$. □

We now present an overview of the analysis, leading to a proof of Lemma 3. The main article of interest is the following lemma that bounds the variance of X. Let n denote the number of items that have mapped to the level l.

Lemma 4. *Suppose that $n \le \frac{K}{4}$, $\epsilon \le \frac{1}{4}$ and $d \ge \max(3+e, 2\log K)$. Then, $\mathrm{Var}_d\left[X\right] \le 2Kq + 2\epsilon^4 K^2 q^2 + K^{-2}$.*

Proof. $\Pr\left\{x_i = 1\right\}$ can be calculated to be $\frac{n(K-1)^{n-1}}{K^n}$. For a fixed pair of distinct buckets, i and j, let $\phi = \phi(i, j)$ denote the event that after a random experiment, buckets i and j are singleton. Therefore, $\Pr\left\{\phi\right\} = \Pr\left\{x_i = 1 \text{ and } x_j = 1\right\} = \frac{n(n-1)(K-2)^{n-2}}{K^n} = n(n-1)p^2(1-2p)^{n-2} < \Pr\left\{x_i = 1\right\}\Pr\left\{x_j = 1\right\}$. The expression $\Pr\left\{\phi\right\} = n(n-1)p^2(1-2p)^{n-2}$ can be viewed as a polynomial in $p = \frac{1}{K}$. Using the principle of inclusion and exclusion, an expression can be obtained for $\mathrm{Pr}_d\left\{\phi\right\}$ that is identical to the terms of this polynomial from degree 2 through d (inclusive). Let T_d denote the coefficient of p^d in the polynomial and let S_d denote the partial sum from $s = 0$ through $d - 2$, that is, until the term for p^d. Since, the polynomial expression $\Pr\left\{\phi\right\}$ is an alternating sum, $|\Pr\left\{\phi\right\} - S_d| \le |T_d|$ and $|\mathrm{Pr}_d\left\{\phi\right\} - S_d| \le |T_d|$.

Therefore, using triangle inequality and the assumptions that $p = \frac{1}{K}$ and $n \leq \frac{K}{4}$,

$$|\mathrm{Pr}_d\{\phi\} - \mathrm{Pr}\{\phi\}| \leq 2|T_d| = 2^{d-1}n(n-1)p^d \binom{n-2}{d-2} < K^{-4}$$

Further, $\mathrm{Pr}\{\phi\} < q^2$. Therefore, $\mathrm{E}_d[x_i x_j] < q^2 + K^{-4}$.

$$\begin{aligned}\mathrm{Var}_d[X] &= \mathrm{E}_d\left[X^2\right] - (\mathrm{E}_d[X])^2 = \mathrm{E}_d[X] + 2\sum_{i<j}\mathrm{E}_d[x_i x_j] - (\mathrm{E}_d[X])^2 \\ &\leq Kq(1+\epsilon^4) + 2\binom{K}{2}\left(q^2 + K^{-4}\right) - (Kq)^2(1-2\epsilon^4) \\ &\leq 2Kq + 2\epsilon^4K^2q^2 + 2K^{-2} \quad \square\end{aligned}$$

Using Chebychev's inequality and Lemma 4, the following lemma can be shown.

Lemma 5. *If $\epsilon \leq \frac{1}{4}$, $\mathrm{E}_d[X] \geq \frac{8}{\epsilon^2}$, $n \leq \frac{K}{4}$ and $d \geq 4\log K$, then,*

$$\mathrm{Pr}_d\{|X - \mathrm{E}_d[X]| \geq \epsilon \mathrm{E}_d[X]\} \leq \tfrac{1}{8} \ . \qquad \square$$

Our proof obligation is now two-fold, namely, (a) if a level satisfying the above conditions on X and Z are found, and $F_0 \geq \frac{72}{\epsilon^2}$, then, $\mathrm{Pr}_d\{|X - \mathrm{E}[X]| > \epsilon\mathrm{E}[X]\} < \frac{1}{8}$, and, (b) if $F_0 \geq \frac{72}{\epsilon^2}$, then, there is a level l such that $X \geq \frac{12}{\epsilon^2}$ and $Z \geq \frac{7K}{8}$. We consider part (a) first. Lemma 5 assumes that $\mathrm{E}[X] \geq \frac{8}{\epsilon^2}$ and $n \leq \frac{K}{4}$. Assuming that we have found a level satisfying $X \geq \frac{12}{\epsilon^2}$, Lemmas 6 shows that $\mathrm{E}[X] \geq \frac{8}{\epsilon^2}$.

Lemma 6. *If $X \geq \frac{12}{\epsilon^2}$ and $K \geq \frac{512}{\epsilon^2}$ then $\mathrm{E}[X] \geq \frac{8}{\epsilon^2}$.*

Proof. Since $X \geq \frac{12}{\epsilon^2}$, $n \geq \frac{12}{\epsilon^2}$. For $\frac{12}{\epsilon^2} \leq n \leq \frac{K}{4}$, $Kq = n(1-\frac{1}{K})^{n-1}$ is an increasing function of n. Therefore, in this range of n, $Kq \geq \frac{12}{\epsilon^2}(1 - \frac{12/\epsilon^2}{512/\epsilon^2}) > \frac{8}{\epsilon^2}$. $\square$

We now show in Lemma 7 that if we have observed $X \geq \frac{12}{\epsilon^2}$, then, n can be neither too small nor too large, that is, $n \in (1 \pm \epsilon)\mathrm{E}_d[n]$ with reasonable probability.

Lemma 7. *If $\epsilon \leq \frac{1}{4}$, $X \geq \frac{12}{\epsilon^2}$ and $d \geq 2$, then,* $\mathrm{Pr}_d\{|n - \mathrm{E}_d[n]| \geq \epsilon\mathrm{E}_d[n]\} \leq \frac{1}{8}$.

Proof. Since, $X \geq \frac{12}{\epsilon^2}$, $n \geq \frac{12}{\epsilon^2}$. Assuming $d \geq 2$, $\mathrm{Var}[n] \leq \mathrm{E}[n]$. If $\mathrm{E}[n] \geq \frac{8}{\epsilon^2}$, then, $\mathrm{Pr}\{|n - \mathrm{E}_d[n]| \geq \epsilon\mathrm{E}_d[n]\} < \frac{\mathrm{Var}[n]}{(\epsilon\mathrm{E}_d[n])^2} \leq \frac{1}{\epsilon^2\mathrm{E}_d[n]} = \frac{1}{8}$. Otherwise, $\mathrm{Pr}\left\{n \geq \frac{12}{\epsilon^2}\right\} \leq \frac{\mathrm{E}_d[n]}{(12/\epsilon^2 - \mathrm{E}_d[n])^2} \leq \frac{8/\epsilon^2}{16/\epsilon^4} \leq \frac{1}{32}$. $\square$

We now return to the proof obligation of $n \leq \frac{K}{4}$. This actually follows (with reasonable probability) from the observation that Z, the number of non-empty buckets, is at least $\frac{7K}{8}$. The intuitive argument is that as n becomes a larger proportion of K, the number of empty buckets observed must drop. Lemma 8 shows this fact using the following notation.

Let y_i be an indicator variable that is 1 if bucket i is non-empty and is 0 otherwise. Let $Y = \sum_{i=0}^{K-1} y_i$ denote the number of non-empty buckets. Then, $r = \mathrm{Pr}\{y_i = 1\} = 1 - \left(1 - \frac{1}{K}\right)^n$ and $\mathrm{E}[Y] = Kr$. Further, using the arguments above, $|\mathrm{Pr}_d\{y_i = 1\} - r| \leq \epsilon^4 r$.

Lemma 8. *Suppose $\epsilon \le \frac{1}{4}$, $n \ge \frac{K}{4}$ and $K \ge \frac{512}{\epsilon^2}$, then,* $\Pr_d\left\{Y \le \frac{K}{8}\right\} \le \frac{1}{8}$.

Proof. Since $n \ge \frac{K}{4}$, $r \ge 1 - (1 - \frac{1}{K})^{\frac{K}{4}} \ge \frac{3}{4}$. Arguing similarly as in Lemma 4, $\mathrm{Var}_d\left[Y_d\right] \le 2Kr + 2\epsilon^4 K^2 r^2 + 2K^{-2}$. Further, if $n \ge \frac{K}{4}$, then, $\mathrm{E}\left[Y\right] = Kr \ge \frac{3K}{4}$. The lemma now follows by applying Chebychev's inequality. □

Using the union bound to add the error probabilities in Lemmas 5, 7 and 8, we obtain that if $X \ge \frac{12}{\epsilon^2}$ and $Z \ge \frac{7K}{8}$, then, $\mathrm{E}\left[X\right] \ge \frac{8}{\epsilon^2}$, $X \in (1 \pm \epsilon)\mathrm{E}\left[X\right]$ and $n \le \frac{K}{4}$, with probability at least $1 - \frac{1}{8} - \frac{1}{8} - \frac{1}{8} > \frac{5}{8}$. We now consider the second part of the proof obligation, namely, to show that there is a reasonable probability of finding a level l such that $X \ge \frac{12}{\epsilon^2}$ and $Z \ge \frac{7K}{8}$, provided, $F_0 \ge \frac{72}{\epsilon^2}$.

Lemma 9. *Suppose $\epsilon \le \frac{1}{8}$, $K = \frac{720}{\epsilon^2}$, $F_0 \ge \frac{72}{\epsilon^2}$ and $l = \lceil \log \frac{F_0}{36/\epsilon^2} \rceil$. Then, $Z \ge \frac{7K}{8}$ and $\frac{12}{\epsilon^2} \le X \le \frac{81}{\epsilon^2}$ is satisfied with probability $\frac{7}{8}$.*

Proof. The expected number of items that map to level l is $\mathrm{E}_d\left[n\right] = \frac{F_0}{2^l}$ which lies in the range of $\frac{36}{\epsilon^2}$ and $\frac{72}{\epsilon^2}$. Arguing as in Lemma 7, it follows that $\Pr_d\left\{|n - \mathrm{E}_d\left[n\right]| \ge \epsilon n\right\} < \frac{1}{24}$. Therefore, we have $\frac{36}{\epsilon^2}(1-\epsilon) \le n \le \frac{72}{\epsilon^2}(1+\epsilon)$, with probability at least $\frac{23}{24}$. Therefore $Z \ge K - \frac{81}{\epsilon^2} = \frac{720}{\epsilon^2} - \frac{81}{\epsilon^2} > \frac{7K}{8}$. Further, $\mathrm{E}\left[X\right] = Kq = n(1-\frac{1}{K})^{n-1} \ge n(1-\frac{n}{K}) > \frac{30}{\epsilon^2}$ and $X \le n \le \frac{72(1+\epsilon)}{\epsilon^2} = \frac{81}{\epsilon^2}$. Therefore, by Chebychev's inequality, it follows that,

$$\Pr_d\left\{X \le \tfrac{12}{\epsilon^2}\right\} \le \frac{\mathrm{Var}_d[X]}{(\mathrm{E}_d[X] - \frac{12}{\epsilon^2})^2} \le \frac{2Kq + 2\epsilon^4 K^2 q^2 + K^{-2}}{(Kq(1-\epsilon^4) - \frac{12}{\epsilon^2})^2} \le \frac{1}{16} \ .$$

Adding the error probabilities of $\frac{1}{16} + \frac{1}{24}$, we obtain the lemma. □

Proof (Of Lemma 1). If $F_0 \ge \frac{72}{\epsilon^2}$, then, by Lemma 9, there exists a level with probability at least $\frac{7}{8}$ such that $\frac{12}{\epsilon^2} \le X \le \frac{81}{\epsilon^2}$ and $Z \ge \frac{7K}{8}$, and in this case, the algorithm returns a non-$\perp$ value as the estimate of F_0. Recall that we keep $96 \cdot \log \frac{2}{\delta}$ independent copies of the basic data structure. By Chernoff bounds, the probability that the number of copies that returns a non $\perp$ value is less than $60 \cdot \log \frac{2}{\delta}$ is at most $\frac{\delta}{8}$.

Therefore, with probability at least $1 - \frac{\delta}{2}$, we have at least $60 \cdot \log \frac{2}{\delta}$ estimates for which a level with the required properties is found. By previous arguments, for each such estimate, the probability that $X \in (1 \pm \epsilon)\mathrm{E}\left[X\right]$ is at least $1 - \frac{11}{28} = \frac{17}{28}$. By Lemma 3, in each of these cases, $\hat{F}_0 \in (1 \pm 2\epsilon)F_0$. Therefore, by a standard application of Chernoff's bounds of returning the median of $60 \log \frac{2}{\delta}$ estimates $\hat{F}_0$, the error probability is reduced to $\frac{\delta}{2}$. Adding the error probabilities $\frac{\delta+\delta}{2} = \delta$, we obtain the statement of the lemma. □

4 Conclusions

We present novel, time and space efficient algorithms for estimating F_0 over data streams with insertion and deletion operations.

Acknowlegement

The author thanks Barna Saha for comments on the paper.

References

1. Noga Alon, Yossi Matias, and Mario Szegedy. "The Space Complexity of Approximating the Frequency Moments". In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing STOC, 1996*, pages 20–29, Philadelphia, Pennsylvania, May 1996.
2. Noga Alon, Yossi Matias, and Mario Szegedy. "The space complexity of approximating frequency moments". *Journal of Computer Systems and Sciences*, 58(1):137–147, 1998.
3. Ziv Bar-Yossef, T.S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. "Counting distinct elements in a data stream". In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2002*, Cambridge, MA, 2002.
4. Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. "Min-wise independent permutations (Extended Abstract)". In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing STOC, 1998*, pages 327–336, Dallas, Texas, May 1998.
5. Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. "Min-wise independent permutations". *Journal of Computer Systems and Sciences*, 60(3):630–659, 2000.
6. Devdatt Dubhashi, Volker Priebe, and Desh Ranjan. "Negative Dependence through the FKG Inequality". Basic Research in Computer Science, Report Series, BRICSRS-96-27.
7. Philippe Flajolet and G.N. Martin. "Probabilistic Counting Algorithms for Database Applications". *Journal of Computer Systems and Sciences*, 31(2):182–209, 1985.
8. Sumit Ganguly, Minos Garofalakis, and Rajeev Rastogi. "Processing Set Expressions over Continuous Update Streams". In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, San Diego, CA, 2003.
9. Sumit Ganguly, Minos Garofalakis, Rajeev Rastogi, and Krishna Sabnani. "Streaming Algorithms for Robust, Real-Time Detection of DDoS Attacks". Bell Laboratories Technical Memorandum, 2004.
10. Philip B. Gibbons. "Distinct Sampling for Highly-accurate Answers to Distinct Values Queries and Event Reports". In *Proceedings of the 27th International Conference on Very Large Data Bases*, Roma, Italy, September 2001.
11. Philip B. Gibbons and Srikant Tirthapura. "Estimating simple functions on the union of data streams". In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2001*, pages 281–291, Heraklion, Crete, Greece, July 2001.
12. Philip B. Gibbons and Srikant Tirthapura. "Distributed streams algorithms for sliding windows". In *Proceedings of the 14th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2002*, pages 63–72, Winnipeg, Manitoba, Canada, August 2002.
13. Piotr Indyk and David Woodruff. "Tight Lower Bounds for the Distinct Elements Problem". In *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC), 2003*, San Diego, CA, 2003.