Deterministic K-Set Structure

Sumit Ganguly and Anirban Majumder
{sganguly,anirban}@cse.iitk.ac.in

IIT Kanpur, India

Abstract. A k-set structure over data streams is a bounded-space data structure that supports stream insertion and deletion operations and returns the set of (item, frequency) pairs in the stream, provided, the number of distinct items in the stream does not exceed k; and returns NIL otherwise. This is a fundamental problem with applications in data streaming [24], data reconciliation in distributed systems [22] and mobile computing [28], etc. In this paper, we study the problem of obtaining deterministic algorithms for the k-set problem.

1 Introduction

Consider scenarios where entities with identity arrive and depart in a critical zone, for example, persons with RF-tags, TCP connections to a given site, etc. The problem is to very efficiently answer the following query:" Are there at most k distinct entities (e.g., persons or items or IP-addresses) in the critical zone, and if so, what are their identities?" Clearly, if there is enough memory to track all the entities, then, an $O(n)$ space solution is the most obvious one. The problem can be effectively solved using a k-set data structure, which is a data structure that (a) supports insertions and deletions of items in a stream or a multi-set, and, (b) supports a *Retrieve* operation that returns all the distinct items and their number of occurrences in the multi-set, provided, the number of distinct items is at most k; and returns NIL otherwise.

Applications of k-set structure arise in diverse areas, ranging from, practical applications in data streams and distributed computing [22, 28] to puzzles in recreational mathematics [5, 24]. In a distributed computing scenario, a host and a PDA may proceed asynchronously with their computations due to low (or non-existent) communication bandwidth between them. Later a reconciliation mechanism is needed to synchronize a specific collection of bits between the two hosts. Set reconciliation [22, 28] can be used as a mechanism that minimizes the communication of bits between the two hosts. We discuss the set reconciliation problem in Section 2—here, we note that the k-set structure can be used to give a space-optimal solution to the set and multi-set reconciliation problems. In a distributed computing environment, the k-set structure can be used for reconciling changes to shared structures, such as files, transaction logs, etc., with the minimum communication necessary.

1.1 Data Streaming Model

We use data streaming as the model for specifying the k-set structure. A data stream $\mathcal{S}$ is viewed as a sequence of records of the form (i, v), where, i is the identity of the data item that is assumed to belong to the domain $\mathcal{D} = \{1, 2, \ldots, N\}$ and v is the change in the frequency of i. For simplicity, we assume that v is integral, where, a positive value of v corresponds to v insertions of i, and a negative value of v corresponds to v deletions of i. The frequency f_i of an item i is defined as the sum of the changes in the frequencies of i, that is, $f_i = \sum_{(i,v)\in\mathcal{S}} v$. At any given time, the multi-set corresponding to the stream is defined as $\{(i, f_i) \mid f_i \neq 0\}$. Let M denote an upper bound on the absolute value of the frequency of an item in the stream, that is, $M \geq f_i$, for each $i \in \mathcal{S}$.

Data streams that allow insertions and deletions of items but allow item frequencies to be both positive and negative are referred to as *general* update streams. A special case of the model arises when item frequencies in the data stream are always constrained to be non-negative. Such streams are referred to as *strict* update streams. In [24], Muthukrishnan refers to the two models as the turnstile model and the strict turnstile model, respectively. Computations over non-negative update streams are studied in [3, 6, 8, 12, 17, 18]. The general update streaming model has been used to detect changes in streams [7, 27].

1.2 *K*-set structure

We now formally two variants of the *k-set structure*, called *strong* and *weak* k-sets, respectively.

Definition 1. *A strong k-set structure over a data stream is a structure that supports the following three operations, (a) procedure* Update*, for updating the data structure corresponding to stream insertion and deletion operations, (b) procedure* Count*, that returns the number n of items with non-zero frequencies, given that $n \leq k$, (c) procedure* Retrieve*, that returns the multi-set S of (i, f_i) pairs in the stream with $f_i \neq 0$, provided, $n \leq k$, and, (d) (c) procedure* IsCard *(Is Cardinality at most k?), that returns* TRUE *if the number of distinct items in the stream (multi-set) is at most k and returns* FALSE *otherwise. A weak k-set supports all the above procedures, except, procedure* IsCard*.* □

K-set structures are readily designed by using classical dictionary structures including heaps, binary search trees, red-black trees, AVL trees, hash tables, etc., that store the entire set S of items, and therefore, require $\Omega(|S|) = \Omega(N)$ bits in the worst case. However, unlike the generality of dictionary structures, a k-set structure only needs to return the set of items provided this set is of size at most k. Therefore, there is a possibility of solving this problem in significantly lower space; as reinforced by the following lower bound argument. There are $\binom{N}{k}(2M-1)^k$ possible multi-sets of of size k over the domain $\{1, 2, \ldots, N\}$ such that $|f_i| \leq M$

and $f_i \neq 0$. Each such multi-set must map to a distinct memory pattern of a deterministic algorithm (otherwise, the algorithm makes an error in at least one of the inputs). Therefore, a deterministic k-set structure requires $\Omega(\log(\binom{N}{k}(2M)^k)) = \Omega(k(\log \frac{N}{k} + \log M))$ bits of space. We are interested in obtaining designs that use space that is close to this lower bound.

1.3 Previous work on Randomized k-set structures

A randomized strong k-set structure is a structure that uses random bits in the execution of the procedures Update, Retrieve and IsCard. Further, the Retrieve operation returns all the n distinct items in the set when $n \leq k$ with high probability, and the *IsCard* operation returns TRUE or FALSE correctly, with high probability. A randomized version of a weak k-set structure can be specified similarly.

Several constructions of randomized k-set structures are known. The COUNTSKETCH algorithm [4], extended by [7] to handle deletion operations, can be used to obtain a randomized strong k-set structure. This structure uses space $O(k(\log \frac{k}{\delta})(\log M + \log N)^2 \log(\log M + \log N) \log M)$ bits. An alternative strong k-set structure can be constructed using the MAJORITY-based data structure [8] and the COUNT-MIN sketch algorithm [6, 24]. Such a construction uses space $O(k \log \frac{k}{\delta}(\log M + \log N) \log M)$ bits. The randomized k-set structure [10] uses $O(k(\log M + \log N) \log \frac{k}{\delta})$ bits and is currently the most efficient randomized k-set structure. The above three techniques are applicable to the general update streaming model (i.e., $f_i \lesseqgtr 0$). Muthukrishnan [24] (Theorem 15) describes a deterministic algorithm from [11] with space complexity $O(k^2 \log^2 N \log M \log^2 k)$ for identifying all top-k items (i.e., $f_i > \frac{\sum_j f_j}{k+1}$), assuming that there are at most k items in the stream. Given that there are at most k items in the stream, a weak k-set structure can be used to retrieve all the items and their frequencies.

1.4 Contributions

We study the problem of designing k-set structures and present deterministic k-set structures for the variants of the problem. Our results are the following. For the strict update streaming model, we present a near-optimal space construction for strong k-sets using $O(k(\log M + \log N)^2)$ bits. The time complexity of procedures *Retrieve*, *IsCard* and *Count* are $O(k^4(\log M + \log N)^2)$, $O(k^3)$ and $O(k^3)$ respectively. For the general update model, we present a weak k-set construction that uses $O(k(\log N + \log M + \log s))$ bits to implement procedure *Count* and $O(k^2(\log M + \log N))$ bits to implement procedure *Retrieve*, where, s is the sum of the absolute values of the updates to the stream, that is, $s = \sum_{(i,v) \in \text{ stream}} |v|$. Alternatively, procedure *Count* can also be implemented using $O(k^2 \log N + k \log M)$ bits. We show that procedure *IsCard* requires $\Omega(N)$ bits in the general update streaming model.

1.5 Organization

The remainder of the paper is organized as follows. We discuss related problems in Section 2. Section 3 presents space lower bounds for strong k-sets. We present our k-set structure in Section 4 and discuss an implementation using finite fields. In Section 5, we discuss an implementation using real arithmetic with finite precision. Section 6 presents our experimental results. Finally, we conclude in Section 7.

2 Related Problems

The k-set problem has been posed and used earlier in different forms and in different applications. We present two such examples, one from mobile communications (PDA synchronization) [22, 28] and another from recreational mathematics [24].

2.1 Set Reconciliation Problem

Set reconciliation [22, 28] is motivated by distributed systems where multiple hosts compute asynchronously in the face of unavailable and/or low-bandwidth network connectivity by temporarily sacrificing consistency. These problems arise in mobile computing [28], distributed databases and distributed file systems [26, 9], etc.. Such systems typically require some mechanism for repairing whatever inconsistencies are introduced and set reconciliation is one of the mechanisms proposed for this problem. The problem is formalized as follows: given a pair of hosts A and B, each with a set of items S_A and S_B respectively from the domain $\mathcal{D} = \{1, 2, \ldots, N-1\}$, what is the minimal amount of communication (in terms of numbers of bits exchanged and the numbers of rounds of messages) such that both A and B are able to determine the union of their sets. The problem is to design solutions that require a communication complexity close to $O(k \log N)$, where, k is a known upper bound on the size of the symmetric difference between the sets of the two hosts, that is, $k \geq |S_A - S_B| + |S_B - S_A|$).

The set reconciliation problem can be easily solved using a weak k-set structure. The host A inserts all its items with frequency 1 into the k-set structure and sends it to B. B deletes all its items from the k-set, and invokes the *Retrieve* function of the k-set to retrieve the identity of the items. Since, the space complexity of a k-set structure is $O(k \log N)$ (in this case, $M = 1$), this gives an optimal communication complexity for the problem.

The work in [22] presents the following elegant technique to solve this problem. For simplicity, assume that $S_B \subset S_A$. First, A and B locally construct their respective characteristic polynomials.

$$f_A(z) = \prod_{u \in S_A} (z - u) \text{ and } f_B(z) = \prod_{u \in S_B} (z - u)$$

Then,

$$\frac{f_A(z)}{f_B(z)} = \frac{f_{A-B}(z)}{f_{B-A}(z)} \tag{1}$$

Next, the polynomial f_A is evaluated at k points (the authors [22] use the points $a_1 = -1, a_2 = -2, \ldots, a_k = -k$), not belonging to the domain $\mathcal{D}$ to obtain a vector (or a transform) $F_A = [f_A(-1), \ldots, f_A(-k)]$. The transform F_A is then transmitted to B. In an analogous way, B computes its characteristic polynomial and evaluates the polynomial at the *same* k distinct points, $a_1, a_2, \ldots, a_k$. By equation (1) (recall that for simplicity, we have assumed that $B \subset A$), we can calculate the transform F_{A-B} as follows.

$$F_{A-B} = \left[\frac{f_A(a_1)}{f_B(a_1)}, \cdots, \frac{f_A(a_k)}{f_B(a_k)}\right]$$

Since, $|A-B|$ is given to be of size at most k, the polynomial $f_{A-B}(z)$ is of degree at most k. The problem is now to invert the transform F_{A-B} to obtain the polynomial f_{A-B}, which can be easily done using standard interpolation techniques. Since, $f_{A-B}(z) = \prod_{u \in S_A - S_B}(z-u)$, the factors of f_{A-B} give the items in $A-B$. A single pass over the domain (field) is sufficient to retrieve all the factors. For a more efficient algorithm for this problem, see Section 4.2.

The authors [22] note that it is sufficient to perform the polynomial computations over *a finite field of size* $q \geq N$ (for simplicity, assume that N is a power of 2 and the finite field used is $GF(N)$ of characteristic 2). Therefore, each member of the transform F_A, namely, $f_A(u_j)$, requires $\log N$ bits to be represented exactly. The total number of bits transmitted is the size of F_A, which is bounded by $k \log N$ bits.

2.2 Multi-Set Reconciliation Problem

From the discussion above, it might appear that a k-set structure can be designed based on the solution to the set reconciliation problem presented in [22]. However, the polynomial interpolation technique presented above does not present an efficient solution to the following *multi-set* reconciliation problem. In the multi-set reconciliation problem, there are two hosts A and B, each having two multi-sets T_A and T_B, where, the items are from the domain $\{0, 1, \ldots, N-1\}$ and the frequency of any item is at most M and is non-negative. The problem is to use the minimum number of communication bits and/or rounds of communication so that each host knows the other's multi-set.

We first show that the above problem is easily solved using a k-set structure. Suppose that we have an upper bound k on the number of distinct items that do not have the same frequencies in the two multi-sets. Host A inserts its multi-set (i.e., (item, frequency) pairs) into a weak k-set and then transmits the k-set to B. B deletes its multi-set from the k-set and then retrieves the items and their frequencies by invoking procedure *Retrieve*. Items with

positive (resp. negative frequencies) have a higher frequency in the multi-set for B (resp. A) than in A (resp. B). This approach requires $O(k(\log M + \log N))$ bits of communication from A to B.

The characteristic polynomial interpolation method can be adapted to solve the multi-set reconciliation problem as follows. Let f_i and g_i denote the frequency of item i (number of occurrences) in M_A and M_B, respectively. Then, the characteristic polynomials corresponding to the multi-sets M_A and M_B are as follows.

$$h_A(z) = \prod_{i \in M_A} (z - i)^{f_i}, \quad h_B(z) = \prod_{j \in M_B} (z - j)^{g_j}$$

Therefore,

$$\frac{h_A(z)}{h_B(z)} = \frac{\prod_{i:f_i > g_i} (z - i)^{f_i - g_i}}{\prod_{j:g_j > f_j} (z - j)^{g_j - f_j}}$$

For simplicity, assume that $M_B = \phi$ and there are at most k distinct items in M_A. The corresponding polynomial $h_A(z) = \frac{h_A(z)}{h_B(z)}$ can have degree kM. In general, in order to find the coefficients of a degree m polynomial, it is necessary to maintain its value at m distinct points. The transform (or interpolation)-based procedure therefore has a space complexity of $O(kM \log N)$ bits.

2.3 Missing Numbers Puzzle

Muthukrishnan [24] presents the "Missing numbers puzzle" as a simple abstraction of a problem over data streams. In the missing numbers puzzle, there are two parties, namely, Paul and Carole. Paul sends an arbitrary permutation of numbers from 1 to N, except at most k of these numbers, to Carole. Carole is unaware of the permutation used by Paul. The problem for Carole is to find the missing numbers. Clearly, if Carole has N bits of memory, then she can trivially solve the problem by using it to remember all the numbers presented to her. Therefore, the problem for Carole really is to find the missing numbers *using as few bits as possible*. As pointed out by [24], this problem is an abstraction of problems in data streaming.

[24] presents simple solutions for the case when there are one or two numbers are missing. A weak k-set structure easily solves the missing numbers problem, when there are at most k items that are missing. Initially, Carole inserts all numbers 1 to N into a k-set, each with frequency 1. Next, for every number i that is supplied by Paul, Carole decrements the frequency of i by 1, effectively, deleting i from the current set. The remaining set of items is exactly the set of missing numbers.

3 Lower Bounds

In this section, we discuss space lower bounds for k-set structure. As outlined in the introduction, a space lower bound of $\Omega(k(\log N + \log M - \log k))$ can be easily shown for the k-set structure. The argument is effective both for the streaming model where items have non-negative item frequencies, and for the model where item frequencies can be both positive or negative. We now consider space lower bounds for procedure *IsCard*, namely, the problem of testing deterministically whether there are k or less distinct items in the stream.

Lemma 1. *Consider a streaming model where item frequencies can be both positive or negative. Then, for any $1 \leq k \leq \frac{N}{4}$, a deterministic algorithm for testing whether the number of distinct items in the stream is at most k requires $\Omega(N)$ bits.*

Proof. Let $\mathcal{S}$ be a family of sets of size $\frac{N}{2}$ such that the distance between any pair of sets in the family is at least $\frac{N}{4}$. Using simple counting techniques, it is easy to show that there exists such families with size $2^{\Omega(N)}$.

Let S and T be two such sets from the family. Construct two streams from S and T respectively where the item frequency is 1 for each element in the corresponding set. Consider the memory patterns of a k-set structure after processing the streams independently. We claim that the two memory patterns must be different for the following reason. Consider a sequence of deletions of $\frac{N}{2} - k$ items from S that leaves S with k items. Since, S and T have at most $\frac{N}{4}$ items in common, the same sequence of deletions applied to T leaves T with at least $\frac{N}{2} - k$ distinct items (in which at least $\frac{N}{4} - k$ have negative frequencies). It follows that the space required by the k-set structure is at least $\Omega(\log|\mathcal{S}|) = \Omega(N)$. □

Lemma 1 justifies the terminology of weak and strong k-sets.

4 *K*-set structure

In this section, we present our design of a k-set structure. We keep $s = 2k + 2$ counters, denoted by $l_0, l_1, \ldots, l_{2k+1}$ that track the following quantities.

$$l_r = \sum_{x_i \in \text{ stream}} f_i x_i^r, \quad r = 0, 1, \ldots, 2k+1 \tag{2}$$

The counters can be easily updated in the face of insertions and deletions occurring in the stream. For every update (x_i, v) occurring in the stream, we update the r^{th} counter as follows:

$$l_r := l_r + v \cdot x_i^r, \quad \text{for } r = 0, 1, \ldots, 2k+1 \ .$$

We use the following notation in this section. Given n distinct items $x_1, x_2, \ldots, x_n$, each of which lies in the interval $1 \leq x_i \leq N$, we let $X = X(n)$ denote the $n \times n$ diagonal matrix

that has x_i in its i^{th} diagonal entry and zeros elsewhere. Given a set of n frequency values, $f_1, f_2, \ldots, f_n$, we let F denote the diagonal matrix whose i^{th} diagonal entry is f_i and is zero elsewhere. Let f be the column vector $[f_1, f_2, \ldots, f_n]^T$. That is,

$$X = \begin{bmatrix} x_1 & & & \\ & x_2 & & \\ & & \ddots & \\ & & & x_n \end{bmatrix} \qquad F = \begin{bmatrix} f_1 & & & \\ & f_2 & & \\ & & \ddots & \\ & & & f_n \end{bmatrix} \qquad \text{and } f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} .$$

For $1 \leq n \leq k$ and $0 \leq r \leq 2k - n$, let $V(r, n)$ denote the $n \times n$ matrix shown below. For a given set of values $x_1, x_2, \ldots, x_n$, for brevity, we refer to $V(0, n)$ as V, as follows.

$$V(r,n) = \begin{bmatrix} x_1^r & x_2^r & \cdots & x_n^r \\ x_1^{r+1} & x_2^{r+1} & \cdots & x_n^{r+1} \\ & \vdots & \vdots & \\ x_1^{r+n-1} & x_2^{r+n-1} & \cdots & x_n^{r+n-1} \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ & \vdots & \vdots & \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{bmatrix}$$

The following identity is a direct consequence of the definition.

$$V(r,n) = VX^r \tag{3}$$

Let $w(s, r)$, B_r and C_r denote the following $r \times 1$ column matrix and $r \times r$ square matrices respectively.

$$w(s,r) = \begin{bmatrix} l_s \\ l_{s+1} \\ \cdots \\ l_{r+s-1} \end{bmatrix} \qquad B_r = \begin{bmatrix} l_0 & l_1 & \ldots & l_{r-1} \\ l_1 & l_2 & \ldots & l_r \\ \vdots & \vdots & \vdots & \vdots \\ l_{r-1} & l_r & \ldots & l_{2r-2} \end{bmatrix}, \text{ and } \qquad C_r = \begin{bmatrix} l_1 & l_2 & \ldots & l_r \\ l_2 & l_3 & \ldots & l_{r+1} \\ \vdots & \vdots & \vdots & \vdots \\ l_r & l_{r+1} & \ldots & l_{2r-1} \end{bmatrix} . \tag{4}$$

4.1 Basic Properties

The main property of this structure is summarized in Lemmas 2 and 3. Lemmas 2 holds for streaming models in which item frequencies could be both positive and negative.

Lemma 2. *Suppose that there are $n \leq k$ distinct items in the stream. Then, (a) rank$(B_{k+1}) = n$, (b) the items $x_1, x_2, \ldots, x_n$ are the eigenvalues of the matrix $B_n^{-1}C_n$ and, (c) the frequency vector is given by $f = V^{-1}w(0, n)$.*

Proof. Suppose there are $n \leq k$ distinct items $x_1, x_2, \ldots, x_n$ in the stream, where, each $x_i \in \{0, 1, 2, \ldots, N-1\}$, for $i = 1, 2, \ldots, n$. Let $V(r) = V(r, n)$ and $w(r) = w(r, n)$, for $r = 0, 1, 2, \ldots, 2k + 1 - n$. Thus, equation (2) can be rewritten as follows.

$$V(r)f = VX^r f = w(r), \quad r = 0, \ldots, 2k + 1 - n \ .$$

Since, the x_i's are non-zero and distinct, $V(r) = VX^r$ is invertible for each value of $0 \leq r \leq 2k+1-n$. Therefore,

$$VXV^{-1}w(r) = VX^{r+1}((VX^r)^{-1}w(r)) = VX^{r+1}f = w(r+1), \quad 0 \leq r \leq 2k+1 \ .$$

Let A denote the matrix VXV^{-1}. The above set of equations can be expressed as

$$AB_n = C_n \tag{5}$$

Since A is in the eigen-decomposition form, X is the diagonal eigenvalue matrix of A. In other words, the distinct items $x_1, \ldots, x_n$ are the eigenvalues of the matrix A'. Further,

$$\begin{aligned} B_n &= [w(0)\ w(1)\ \ldots\ w(n-1)] = [Vf, VXf, VX^2f, \ldots, VX^{n-1}f], \\ &= V[f,\ Xf,\ X^2f,\ \ldots\ , X^{n-1}f] = VFV^T \end{aligned} \tag{6}$$

Since, V is invertible and none of the f_i's are 0, B_n is invertible, and therefore has rank n.

Since B_n is the left $n \times n$ sub-matrix of B_{k+1}, $\text{rank}(B_{k+1}) \geq \text{rank}(B_n) = n$. Let U denote the $(k+1) \times n$ matrix $V(0, k+1)$ and let $U(j)$ be the $(k+1) \times n$ matrix $V(j, k+1)$, for $0 \leq j \leq k$. Therefore, (2) can be equivalently written as: $U(j)f = UX^j = w(j, k+1)$, for $j = 0, \ldots, k$. Thus,

$$B_{k+1} = [w(0, k+1), w(1, k+1), \ldots, w(k, k+1)] = U[f, Xf, \ldots, X^k f] = UFU^T \ .$$

Since, U and F each have rank n, it follows that $\text{rank}(B_{k+1}) \leq n$. As shown earlier, $\text{rank}(B_{k+1}) \geq n$. Therefore, $\text{rank}(B_{k+1}) = n$. □

Lemma 2 can be used to design procedures *Retrieve* and *Count* for strong and potent k-sets respectively, provided it is known that the number of distinct elements in the stream is at most k. Lemma 2(a) and Lemma 3 provides the basis for testing whether a stream has k or less distinct items. Notably, however, Lemma 3 is applicable only for streams where item frequencies are all non-negative (or, analogously, all non-positive).

Lemma 3. *For strict update data streams (i.e., $f_i \geq 0$, for all i) there are $n > k$ distinct items in the stream with positive frequencies if and only if rank$(B_{k+1}) = k+1$.*

Proof. The if part is the statement of Lemma 2(a). Suppose there are $n > k$ distinct items with positive frequencies. Let G denote the diagonal matrix with $G_{i,i}$ set to the positive square root of f_i. Therefore,

$$B_n = VFV^T = VG^2V^T = (VG)(VG)^T \ .$$

It follows that B_n is a positive definite matrix, and hence, all left most determinants of B_n are positive. Since, $n > k$, in particular, $\det(B_{k+1}) > 0$, and therefore, $\text{rank}(B_{k+1}) = k+1$. □

4.2 Implementing k-sets

In this section, we present a space efficient implementation of k-sets.

Lemma 2 holds for any finite field of characteristic at least $2M$ and having N distinct values. The number $2M$ is chosen to account for positive frequencies $1, \ldots, M$ and negative frequencies, $-M, \ldots, -1$. Choose an appropriate prime number p larger than $2M$ and let d be the smallest integer ≥ 1, such that $p^r > N$. Let F be the field $GF(p^{r+1})$. The elements of F can be naturally represented using $O(\log M + \log N)$ bits. The counters, $l_0, \ldots, l_{2k+1}$ are each maintained as elements over F; the total space requirement is $O(k(\log M + \log N)$ bits.

Suppose it is given that the number of items in the stream $n \leq k$ (i.e., weak k-set), then, n can be found as $\text{rank}(B_{k+1})$. By Lemma 2, this property holds in general for data streams where item frequencies are both positive or negative. The identities of the items with non-zero frequencies can be found as the eigenvalues of A. Since, iteratively convergent methods cannot be used over finite fields, computing the eigenvalues of A over finite fields in general requires the computation of the roots of the characteristic polynomial $F(z) = \det(A - zI) = 0$, which is computationally expensive. For data streams where item frequencies can be both positive and negative, the method of finding eigenvalues over $\mathbb{R}$ given in Section 5 can be applied. If item frequencies cannot be negative, then the following algorithm (based on an application of dyadic intervals) can be used for finding the roots of the characteristic polynomial.

Finding roots of the characteristic polynomial. We now assume that item frequencies cannot assume negative values. Since there are n eigenvalues, the characteristic polynomial $F(z)$ is of the form $F(z) = \alpha \prod_{a \in S}(z - a)$, where, S is the set of items in the stream with non-zero frequency and α is a constant. Let F be a field with characteristic larger than kM.

Instead of maintaining a single set of $2k + 2$ counters, we maintain a collection of $L = \lceil \log |F| \rceil - \lfloor \log k \rfloor + 1$ sets of counters, where each set consists of $2k + 2$ counters. The s^{th} counter set is denoted as $\{l_r^s\}_{r=0,1,\ldots,2k+1}$, for $0 \leq s \leq L - 1$. For $0 \leq s \leq L - 1$, define a family of functions $h_s : \{0, 1, \ldots, 2^d - 1\} \rightarrow \{0, 1, \ldots, 2^{d-l} - 1\}$ as follows.

$$h_0(a) = a \qquad \text{and} \qquad h_s(a) = a \div 2^s$$

where, $a \div 2^s$ is the quotient when the integer a is divided by the integer 2^s. It follows directly that, for any $s \geq 0$ and given value of $c = h_s(a)$, there are exactly two distinct values of b such that $b = h_{s-1}(a')$. Corresponding to each stream update of the form (x, v), we update each of the s counter sets as follows.

$$l_r^s = l_r^s + (h(x))^r v, \quad 0 \leq r \leq 2k + 1, 0 \leq s \leq L - 1 \ .$$

Let $f_a^{(s)}$ denote the frequency of item a at level s. Then, $f_a^{(s)} = \sum_{b:h_s(b)=a} f_b$. If a has positive frequency $f_a > 0$, then, $f_{h_s(a)}^{(s)} > 0$ has positive frequency at level s, for $1 \leq s \leq L - 1$ (vice-versa may not be true). Let n_s denote the number of distinct items with positive frequencies

at level s. Let $A_{n_s}^{(s)}$, $B_{n_s}^{(s)}$ and $C_{n_s}^{(s)}$ respectively denote the corresponding matrices obtained from the counters at level s, for $s = 0, 1, \ldots, L-1$. Let $F_s(z)$ denote the characteristic polynomial of $A_n^{(s)}$, that is, $F_s(z) = \det(A_{n_s}^{(s)} - zI)$. By construction, we have the following property

$$F(a) = 0 \text{ only if } F_s(h_s(a)) = 0 \ .$$

For each value of s starting from L and counting down to 0, we obtain a set of items of size at most $2k$ that are potentially roots of $F_s(z)$. At level L, there are at most k distinct items that are then checked to see if $F_s(a_s) = 0$ (or equivalently, if $A_n^{(s)} - a_s I$ is singular). Therefore, at each level, there cannot be more than k candidates that pass the above test. Each candidate item a at level s corresponds to exactly 2 candidates at level $s-1$; therefore, the number of potential candidates at any level is no more than $2k$. Proceeding in this manner, we obtain the set of items with positive frequencies at level 0.

The data structure maintains $(2k+2)$ counters for at most $\log|F|$ levels. Therefore, its space complexity is $O(k(\log N + \log M)^2)$ bits. Testing whether an item x is an eigenvalue can be done by calculating the rank of $A - xI$, which can be done in time $O(k^3)$. Since there are at most $2k$ candidate items at each level and there are $\log|F| - \log k + 1 = \log M + \log N - \log k$ levels, the time complexity of retrieval is $O(k^4(\log M + \log N))$ field operations. We summarize this discussion in the following lemma.

Lemma 4. *A strong k-set structure can be designed for strict update data streams using $O(k(\log M + \log N)^2)$ bits and operations over a finite field of size $O(kM + N)$. The time complexity for procedure* Retrieve *is $O(k^4(\log M + \log N))$ field operations and the time complexity for procedure* IsCard *is $O(k^3)$. For general update data streams, a weak k-set structure can be designed with the space and time complexity mentioned above.* □

5 *K*-set structure using real arithmetic

Lemma 2 translates the problem of retrieving the elements to that of retrieving the eigenvalues of a certain matrix. Today, optimized procedures for computing eigenvalues of real and complex matrices are widely available via packages such as LAPACK [1], MATLAB [21], MATHEMATICA [2], etc.. This makes it interesting and relevant to analyze the space and time complexity of the procedure for retrieving the elements. In particular, we analyze the space complexity and the number of bits of precision needed to count the number of items with non-zero frequency, and to retrieve the identity and frequency of those items without error. We introduce a new parameter, called s, that is an upper bound on the number of updates to the streams, that is, $s = \sum_{(i,v) \in \text{ stream}} |v|$. The results in this section can be summarized as follows. A strong k-set structure can be maintained using space $O(k^2 \log N + k \log M)$ bits. Procedure *Count* can also be implemented using space $O(k(\log M + \log N + \log s))$

bits. Except for procedure *IsCard*, the other procedures are applicable to the general update streaming model (i.e., $f_i \lessgtr 0$).

For $r = 0, \ldots, 2k+1$, the counter $l_r = \sum_i f_i x_i^r$ can have a value as large as MN^{r+1} and can be maintained using $(r+1)\lceil \log N \rceil + \lceil \log M \rceil$ bits of storage. The space required to store the counters $l_0, \ldots, l_{2k+1}$ can be accounted as follows.

$$\sum_{r=0}^{2k+1} (r\lceil \log N \rceil + \lceil \log M \rceil) = \Theta(k^2 \log N + k \log M) \text{ bits.}$$

Notation. The norm $\|M\|$ of a matrix M denotes the 2-norm of M and is defined as the largest eigenvalue of $M^T M$ in absolute value. The condition number [13, 29] of a matrix M is denoted as $\kappa(M) = \|M\|\|M^{-1}\|$.

5.1 Precision for computing rank using Gaussian elimination.

The procedure *Count* can be implemented by determining the rank of the matrix B_{k+1}. If this is done using the standard method of Gaussian elimination, then, the word size must be extended by the logarithm of the condition number of B_{k+1} [13]. The condition number of B_{k+1} can be shown to be $\kappa(B_{k+1}) = O(N^{4k} M k^2 2^{2k})$. This implies that it is sufficient to extend the word size by $\log \kappa(B_{k+1}) = O(k \log N + \log m)$ bits.

5.2 Precision for computing A

The second matrix computation involves calculating $A = C_n B_n^{-1}$, where, $n \leq k$ is the number of items with non-zero frequencies in the stream. We use a simple variant of the QR-decomposition followed by back-substitution to obtain $\tilde{A}$.

Consider the matrix computation $A = C_n B_n^{-1}$, where $n \leq k$. For simplicity, we drop the suffix n from the $n \times n$ matrices C_n and B_n. Note that A has the following form.

$$A = CB^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \cdots & \alpha_n \end{bmatrix} . \tag{7}$$

where, the last row is denoted as the row vector $\alpha^T = [\alpha_1, \alpha_2, \ldots, \alpha_n]$. Let c_n^T denote the n^{th} row of the matrix C. Equivalently,

$$c_n^T B^{-1} = \alpha^T$$

or that, $\alpha^T B = c_n^T$. Taking transposes, we obtain the equivalent equation, $B^T \alpha = c_n$. Since, B is a symmetric invertible matrix, therefore, α is the unique solution to the equation

$$B\alpha = c_n$$

where, c_n is the n^{th} column of C (since, C is symmetric). We now decompose B as $B = QR$ using the classical QR decomposition algorithm [13, 29]. In this decomposition, Q is an orthonormal matrix and R is an upper triangular matrix satisfying $\|R\| = \|A\|$. Therefore, $B\alpha = c_n$ is equivalent to $QR\alpha = c_n$, or, equivalently, $R\alpha = Q^T c_n$. Since, R is an upper triangular matrix, α is obtained using back substitution. Due to limited precision, the matrix R is calculated as $R + \Delta R$. If floating point calculation is used up to s_2 bits of precision, then, $\|\Delta R\| \leq 2^{-\Omega(s_2)}\|R\|$. Therefore, the possible error $\Delta\alpha$ in the calculation of α is given by

$$\|\Delta\alpha\| \leq \kappa(R)\frac{\|\Delta R\|}{\|R\|}\|\alpha + \Delta\alpha\| \leq 2^{-\Omega(s_2)}\kappa(B)\|\alpha + \Delta\alpha\| \tag{8}$$

It follows that the error term $\|\Delta\alpha\|$ is negligible compared to α if s_2 is $O(\log(\kappa(B)))$.

By Lemma 2, we have $B = VFV^T$. By Gershgorin circle theorem, $\|V\| = O(N^{2n+1})$ and $\|V^T\| = O(N^{2n+1})$. Clearly, $\|F\| \leq M$. Therefore, $\|B\| = O(N^{2n+1}M)$. Further, $\|V^{-1}\| \leq O(N^{n+1})$ and $\|(V^{-1})^T\| = O(N^{n+1})$ (see Appendix B). Therefore, $\kappa(B) = O(N^{4n+2}M)$. Since $n \leq k$, it follows that if $s_2 = O(k\log N + \log M)$, then, from equation (8), the error term $\Delta\alpha$ is negligible. Let ΔA denote the error matrix for A, then, by construction (see equation (7))

$$\Delta A = e_n(\Delta\alpha)^T$$

where, e_n is the $n \times 1$ unit column vector with 1 in row n and 0 elsewhere. Therefore,

$$\|\Delta A\| \leq \|\Delta\alpha\| \leq 2^{-\Omega(s_2)}\kappa(B)\|\alpha\| \ . \tag{9}$$

Since, $\|A\|_F \leq \sqrt{n}\|A\|$, it follows that,

$$\|A\|_F = n - 1 + \|a\| \leq \sqrt{n}\|A\|, \qquad \text{or that,} \qquad \|a\| \leq \sqrt{n}\|A\| \ .$$

Substituting in (9), we have,

$$\|\Delta A\| \leq 2^{-\Omega(s_2)}\sqrt{n}\kappa(B)\|A\|$$

Therefore, A is computed to sufficient precision given the following condition.

$$\begin{aligned} &\text{if} \qquad s_2 = O(\log n + \log\kappa(B)) = O(k\log N + \log M) \\ &\text{then} \qquad \|\Delta A\| \leq N^{-\Omega(k)}M^{-\Omega(1)}\|A\| \ . \end{aligned}$$

We summarize the above discussions in the following lemma.

Lemma 5. *There exists a weak k-set structure for the general update streaming model (i.e., $f_i \lesseqgtr 0$) using real arithmetic with finite precision that uses $O(k^2\log N + k\log M)$ bits of space for maintaining the counters. The procedures* Count *and* Retrieve *can be implemented using $O(k\log N + \log M)$ bits of floating point precision.* □

5.3 Space optimization for strict update streams

In this section, we present a simple space reduction technique for procedures *Count* and *IsCard* for strict update streams, that is, streams where item frequencies are non-negative.

Instead of working with the actual item identifiers, namely, x_i, we use $y_i = x_i^{1/N}$ truncated to s_2 bits after the binary point, where, s_2 is a parameter. Thus, we track the counters

$$l'_r = \sum x_i^{r/N} f_i, \quad \text{for } r = 0, \ldots, 2k+1.$$

Stream updates of the form (i, v) are processed as follows:

$$l'_r := l'_r + x_i^{r/N} v, \quad \text{for } r = 0, \ldots, 2k+1.$$

Let y_i denote $x_i^{1/N}$ and let A', B', C', V' etc., denote the matrices corresponding to A, B, C, V etc., that use the counters l'_r instead of l_r and the values $y_i^j = x_i^{j/N}$ instead of x_i^j, for $0 \leq r, j \leq 2k+1$. For example, the matrices V' and B' are shown below; other matrices are similarly constructed.

$$V' = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ y_1 & y_2 & \ldots & y_n \\ y_1^2 & y_2^2 & \ldots & y_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ y_1^{n-1} & y_2^{n-1} & \vdots & y_n^{n-1} \end{bmatrix} \quad \text{and} \quad B'_r = \begin{bmatrix} l'_0 & l'_1 & \ldots & l'_{r-1} \\ l'_1 & l'_2 & \ldots & l'_r \\ l'_2 & l'_3 & \ldots & l'_{r+1} \\ \vdots & \vdots & \vdots & \vdots \\ l'_{r-1} & l'_r & \ldots & l'_{2r-2} \end{bmatrix}.$$

The test for determining whether there are k or less distinct items is given by whether the rank n of B'_{k+1} is at most k and is analogous to the test of the rank of B_{k+1} as proved in Theorem 2.

Consider the computation of the rank of B'_{k+1}. Prior to the rank computation, we normalize the counters l'_r by dividing them by $2l_0$, provided, $l_0 > 0$. If $l_0 = 0$, then, there are no items in the stream with positive frequency. Hence, we assume without loss of generality that $l_0 > 0$. Suppose that $k+1 \leq \frac{N}{2 \log N}$.

$$\tilde{l}_r = \frac{l'_r}{2l_0} = \frac{1}{2\sum_i f_i} \sum_i x_i^{r/N} f_i \leq \frac{N^{r/N}}{2\sum_i f_i} \sum_i f_i \leq \frac{N^{r/N}}{2} \leq 1$$

since, $N^{r/N} \leq N^{(2k+1)/N} \leq 2^{(2k+1)\log N/N} \leq 2$, by assumption that $k+1 \leq \frac{N}{2\log N}$. Further,

$$\tilde{l}_r = \frac{1}{2\sum_i f_i} \sum_i x_i^{r/N} f_i \geq \frac{1}{2\sum_i f_i} \sum_i f_i = \frac{1}{2}.$$

Thus, $\frac{1}{2} \leq \tilde{l}_r \leq 1$. The family of the matrices obtained from B' and C', using, $\tilde{l}_r$ in place of l'_r is denoted by $\tilde{B}, \tilde{C}$ respectively.

Clearly, with infinite precision, both techniques, namely, working with the matrices B, C etc., and working with the matrices B', C' etc. are equivalent. However, we show that by using fixed point arithmetic using word size of $O(\log m + \log N + \log s)$ bits, the problem of determining the rank of B' can be solved exactly for data streams with at most s updates.

We use the QR-decomposition algorithm on the columns of $\tilde{B}_{k+1}$ for computing its rank [15] using standard fixed point arithmetic, that is, by truncating underflows to 0 and ignoring overflows. Lemma 6 follows from the standard properties of the QR-decomposition procedure and is presented in Appendix A.

Lemma 6. *Suppose that the rank of $\tilde{B}$ is computed using fixed point arithmetic using word size of $s_2 = 32(\log N + \log M + \log s)$ bits before and after the binary point. Further, any value that is smaller (in absolute value) than $\frac{1}{2m^2N^{16}}$ is deemed to be 0. Then, the QR procedure exactly computes the rank of $\tilde{B}$.*

Proof. See Appendix A. □

The main consequence of Lemma 6 is Lemma 7, which states that a k-set structure can be designed for strict update streams using real arithmetic that requires space $O(k(\log M + \log N + \log s))$ bits for implementing procedures *Count* and *IsCard*. However, in this method, procedure *Retrieve* still requires $O(k^2 \log N + \log M)$ bits of space for computing the eigenvalues.

Lemma 7. *There exists a design of a k-set structure for the strict update streaming model based on real arithmetic that uses space $O(k\ (\log M + \log N + \log s))$ bits for implementing procedures* Count *and* IsCard. *The time complexity of these procedures is $O(k^3)$ operations over fixed point numbers with $O(k(\log M + \log N + \log s))$ bits before and after the binary point.* □

5.4 A Las Vegas type optimization

The worst-case space complexity of the k-set structure was analyzed in Theorem 5 as $O(k^2 \log N + k \log m + k \log s)$ bits. The worst-case occurs when the k-items are $N-k, N-k+2, \ldots, N-1$, respectively. This "lack of separation" among the items leads to the worst-case precision requirement of the algorithm. In this section, we present a simple *Las Vegas* type of optimization to increase the gap between the items, thereby, reducing the space requirement in practice.

Consider a family of permutations Π over the domain $\{1, 2, \ldots, N\}$ that are approximately k-wise independent with relative error ϵ. Constructions of permutation families can be found in [20, 25, 14, 16, 19]. A brief overview of work in approximately k-wise independent permutations is presented in Appendix C.

Let π be a randomly chosen permutation from Π. Each item x_i over the stream is first transformed into $\pi(x_i)$, and then inserted into the k-set structure. The *Retrieve* operation retrieves the hash values, $\pi(x_1), \ldots, \pi(x_k)$, that are then inverted to retrieve the original items $x_1, \ldots, x_k$. The choice of the random hash function increases the average gap between the items and prevents an adversary from consistently choosing the worst-case input for the algorithm. The expected number of bits of precision and therefore, the expected space complexity of the Las Vegas variant is lower than the original algorithm. It is called the *Las Vegas* type of optimization, since, the space and time required by the algorithm is a random variable, although, the algorithm deterministically gives the correct answer [23]. Lemma 8 presents upper bounds on the expected space complexity of implementing a k-set structure using this method and is the counterpart to Lemma 5. It quantifies the expected economy in space complexity obtained by using this method over the deterministic method (viz., $O(k^2 + k \log M + k \log s)$ bits versus $O(k^2 \log N + k \log M)$ bits).

Lemma 8. *The* Las Vegas *technique implements a weak k-set structure for the general update streaming model using $O(k^2 + k \log M + k \log s)$ bits on expectation.*

Proof. See Appendix B. □

6 Experimental Study

In this section we present the results of preliminary experiments that we performed to compare the space required by the k-set structure implemented using finite precision arithmetic over reals against its theoretical complexity. Our experiments were performed on Sun E250 Server having Eultra Sparc II, 400 MHz Dual processor with 1GB RAM and 18GB Hard disk with the Solaris 8 operating system using the Matlab software version 6.0.0.88.

The first set of experiments test the worst-case of the k-set solution, (i.e, $x_i = N - k + i$, for $i = 0, 1, 2, \ldots k-1$), for $N = 2^{32}$ and varying k from 8 to 256. The frequencies of items are taken as 1 in this case. The space requirement (= bits of precision $\times$ $k \log N$) for the original and the *Las Vegas* variant of the k-set solution is measured and plotted as a function of k, as shown in Figure 1. The experimental results indicate that the space complexity of the basic k-set structure is sub-quadratic, thereby, showing that in practice, the k-set structure method works significantly better than predicted. Figure 1 also shows the space required by the *Las Vegas* variant (measured as the average over 10 runs, for each value of k), that is, as expected, consistently superior to the original solution.

Our second set of experiments evaluate the accuracy of the frequency estimation by choosing 32-bit frequency values at random and associating each frequency value to a number between $N - k + 1, \ldots, N$ (the worst case input, without initial randomization). Figure 2 presents a graph that measures the accuracy of frequency estimation as function of the

number of bits of precision used. The x-axis is normalized as the number of bits of precision used $\times \frac{16}{k}$. We start with the number of bits of precision that were needed to retrieve the items accurately, and increase the precision until the frequencies are retrieved without error. The graph shows that the frequencies are estimated up to 1.2% without using any extra bits. In all cases, the extra bits of precision that were required to retrieve the frequencies accurately is at most 15% of the number of bits to retrieve the items themselves.

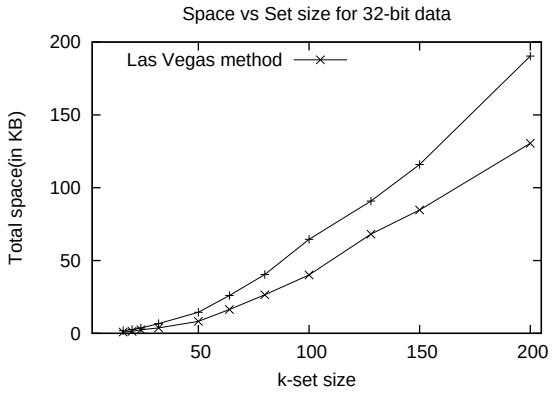

Fig. 1. k-set using eigenvalue computation and Las Vegas method: Space vs k-set size, $N = 2^{32}$.

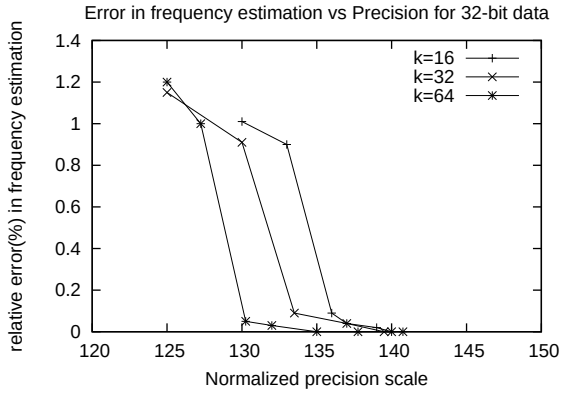

Fig. 2. Accuracy of frequency calculation. Normalized precision = Number of precision bits $\times \frac{16}{k}$

7 Conclusions

We present space and time-efficient, deterministic k-set structures that are nearly space-optimal. The problem of designing more time-efficient k-set structures is open.

References

1. *"LAPACK: The Linear Algebra Package"*. Available from "www.netlib.org/LAPACK".
2. *"MATHEMATICA"*. WolframResearch.
3. 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.
4. M. Charikar, K. Chen, and M. Farach-Colton. "Finding frequent items in data streams". In *Proc. ICALP*, 2002.
5. V. Chauhan and A. Trachtenberg. "Reconciliation puzzles". In *Proc. IEEE GLOBECOM*, 2004.
6. G. Cormode and S. Muthukrishnan. "An improved data stream summary: The Count-Min sketch and its applications". In *Proc. LATIN*, pages 29–38, 2004.
7. G. Cormode and S. Muthukrishnan. "What's New: Finding Significant Differences in Network Data Streams". In *Proc. IEEE INFOCOM*, 2004.
8. Graham Cormode and S. Muthukrishnan. "What's Hot and What's Not: Tracking Most Frequent Items Dynamically". In *Proc. ACM PODS*, pages 296–306, 2003.
9. A.J. Demers, D. H. Greene, C. Hause, W. Irish, and J. Larson. "Epidemic algorithms for replicated database maintenance". In *Proc. ACM PODC*, pages 1–12, August 1987.
10. S. Ganguly. "Counting distinct items over update streams". In *Proc. ISAAC*, pages 505–514, 2005.
11. L. Gasieniec and S. Muthukrishnan. "Deterministic algorithm for estimating heavy-hitters on Turnstile data streams". Manuscript, 2005.
12. Anna Gilbert, Sudipto Guha, Piotr Indyk, Yannis Kotidis, S. Muthukrishnan, and Martin Strauss. "Fast Small-space Algorithms for Approximate Histogram Maintenance". In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC), 2002*, Montreal, Canada, May 2002.
13. Gene H. Golub and Charles F. Van Loan. *"Matrix Computations"*. J. Hopkins Univ. Press, 3rd edition, 1996.
14. W. T. Gowers. "An almost m-wise independent random permutation of the cube". *Combinatorics, Probability and Computing*, 5(2):119–130, 1996.
15. K. M. Hoffman and R. Kunze. *"Linear Algebra"*. Prentice-Hall, 2nd Ed., 1971.
16. S. Hoory, A. Magen, S. Myers, and C. Rackoff. "Simple permutations mix well". pages 770–781.
17. P. Indyk and D. Woodruff. "Optimal Approximations of the Frequency Moments". 2005.
18. Piotr Indyk. "Stable Distributions, Pseudo Random Generators, Embeddings and Data Stream Computation". In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 189–197, Redondo Beach, CA, November 2000.
19. E. Kaplan, M. Naor, and O. Reingold. "Derandomized Constructions of k-Wise (Almost) Independent Permutations". pages 354–365.

20. M. Luby and C. Rackoff. "How to construct pseudorandom permutations and pseudorandom functions". *SIAM J. Comp.*, 17(1):373–386, 1988.
21. The Mathworks, Natick, MA, USA. *"MATLAB 7.1"*.
22. Y. Minsky, A. Trachtenberg, and R. Zippel. "Set Reconciliation with Nearly Optimal Communication Complexity". *IEEE Trans. Inf. Theory*, 49(9):2213–2218, 2003.
23. R. Motwani and P. Raghavan. *"Randomized Algorithms"*. Cambridge University Press, 1995.
24. S. Muthukrishnan. *"Data Streams: Algorithms and Applications"*. Foundations and Trends in Theoretical Computer Science, *Vol. 1, Issue 2*, 2005.
25. M. Naor and O. Reingold. "On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited". *J. Cryptology*, 12(1):29–66, 1999.
26. M. Satyanarayanan. "Scalable, Secure, and Highly Available Distributed File Access". *IEEE Computer*, 23(5), May 1990.
27. R. Schweller, Z. Li, Y. Chen, Y. Gao, A. Gupta, Y. Zhang, P. Dinda, M-Y. Kao, and G. Memik. "Monitoring Flow-level High-speed Data Streams with Reversible Sketches". In *Proc. IEEE INFOCOM*, 2006.
28. D. Starobinski, A. Trachtenberg, and S. Agarwal. "Efficient PDA synchronization". *IEEE Trans. on Mob. Comp.*, 2(1):40–51, 2003.
29. Gilbert Strang. *"Introduction to Linear Algebra"*. Wellesley-Cambridge Press, 2nd ed., 1998.

A Proof of Lemma 6

In this section, we present a proof of Lemma 6. In this proof, the dependence on the size of the stream, namely, $s = \sum_{(i,v)\in \text{ stream}} |v|$, is omitted for simplicity. It is readily incorporated into the statement of the lemma using the following observation. If the precision kept is s_2 bits, then, after s stream updates operations, the error is at most $s2^{-s_2}$. This implies that after s operations, the precision decreases by effectively $s_2 - \log s$ bits. We use this modified value of s_2 as the starting point.

Let $\tilde{B}_{k+1} = [b_0, b_1, \ldots, b_k]$ denote the $k+1$ columns of $\tilde{B}$. If $b_0 = 0$, then, the rank is 0 and the stream is empty. Otherwise, none of the b_i's is the zero vector (since each of the counters l_r is positive). Let $b_j^T \cdot b_i$ denote the inner product of the vectors b_j and b_i. We construct the following sequence of vectors (standard Graham-Schmidt ortho-normalization) using fixed-point arithmetic (that truncates underflows to 0 and ignores overflows) using $s_1 = 32(\log m + \log N)$ bits before and $s_2 = 32(\log m + \log N)$ bits after the binary point.

$$
\begin{aligned}
v_0 &= \frac{b_0}{\|b_0\|} \\
v_i' &= b_i - (v_0^T \cdot b_i)v_0 - (v_1^T \cdot b_i)v_1 - \cdots - (v_{i-1}^T \cdot b_i)v_{i-1} \\
&\qquad\qquad i = 1, 2, \ldots, k \\
v_i &= \frac{v_i'}{\|v_i'\|}, \qquad i = 1, 2, \ldots, k
\end{aligned} \tag{10}
$$

A vector v_i is *deemed to be a non-zero vector* if any of its coordinates has at least one bit which is set to 1 among the bits before the binary point or among the first $1 + \lceil \log \mathcal{D} \rceil$ bits after the decimal point, where, $\mathcal{D} = (k+1)m^2N^{16}$. Otherwise, v_i is *deemed to be the zero vector*. The algorithm keeps a set of vectors v_i which are *deemed to be nonzero* and returns the size of the set(size of the orthogonal basis) as the rank of $\tilde{B}_{k+1}$. The following analysis shows that there are no errors caused by limited precision in the above algorithm.

Analysis of rank computation

For the vector v_i, the algorithm finds the component of b_i along each of the vectors $b_1, b_2, \ldots b_{i-1}$. The component of b_i along v_j is calculated simply as $\frac{v_j^T b_i}{\|v_j\|} v_j$ which is equal to $(v_j^T \cdot b_i)v_j$ since v_j is normalized.By induction, the vectors, $b_0, \ldots, b_{i-1}$ are all independent, otherwise, the algorithm would have terminated earlier. Thus v_i computes the residual component of b_i after subtracting the components along each of $b_0, b_1, \ldots, b_{i-1}$. Assuming infinite-precision, it is clear that if b_i is linearly dependent on the first i vectors, then, after removing its components along those vectors, the residual vector v_i would be zero. Equation (10) can be equivalently written in scalar form (that is, an equation for each of the $k+1$ coordinates of v_i) as follows.

$$(v_i)_j = \tilde{l}_{i+j} - \sum_{r=j}^{j+i-1} \frac{\tilde{l}_r\tilde{l}_{r-j} + \tilde{l}_{r+1}\tilde{l}_{r-j+1} + \cdots + \tilde{l}_{r+k}\tilde{l}_{r-j+k}}{l_{r-j}^2 + l_{r-j+1}^2 + \ldots l_{r-j+k}^2} \cdot \tilde{l}_r, \tag{11}$$

for $j = 1, \ldots, k+1$.

Let $\hat{y}_i$ denote $y_i = x_i^{1/N}$ truncated to s_2 bits after the binary point. Thus, $|\hat{y}_i - y_i| \leq 2^{-s_2}$. Hence $|\hat{y}_i^r - y_i^r| \leq k \cdot N^{\frac{2k}{N}} \cdot 2^{-s_2}$ for r = 0, 1, …, 2k+1 . Let's call this value as δ. Due to truncation after s_2 bits beyond the binary point, each of the counters $\tilde{l}_r$ has an error of at most $k \cdot \delta$. Each multiplication of two of the $\tilde{l}_r$'s introduces an additive error of at most $3k^3 \cdot \delta$. Consider the numerator of any summand of (11). There are a total of $k+1$ multiplications and k additions; thus the total error introduced is at most $3 \cdot k^4 \cdot \delta$. The denominator is at most k and incurs truncation error that is bounded in absolute value by $3 \cdot k^4 \cdot \delta$ (since, there are k multiplications and k additions, each of which incurs an error of at most $3 \cdot k^3 \cdot \delta$). Thus, the total additive error in the calculation of the summand in equation (11) for a given value of r is bounded by $\frac{U+3 \cdot k^4 \cdot \delta}{V-3 \cdot k^4 \cdot \delta} - \frac{U}{V}$, where, U and V are the correct values for the numerator and the denominator for an index r. Since, $\frac{1}{m} \leq U, V \leq (k+1)$, the above error can be bounded in absolute value by $10k^2 \cdot m^2 \cdot \delta$ if $s_2 \geq 6 + 5\log k + \log m + \frac{2k}{N} \log N$. There are $i \leq k+1$ summands in the calculation of $(v_i)_j$; the error due to truncation of $(v_i)_j$ is at most $\epsilon = 10k^3 \cdot m^2 \cdot \delta$. By equation (11), the value of the fraction for each summand index r is at least $\frac{1}{m^2}$. Therefore, the minimum absolute value of $(v_i)_j$, if it is not zero, is at least $\gamma = \frac{1}{m} - \frac{1}{m^2}$. Therefore, by choosing $s_2 \geq 6 + 6\log k + 3\log m + \frac{2k}{N} \cdot \log N$ gives $\gamma > \epsilon$.

The maximum absolute value of $(v_i)_j$ is at most $m(k+1)^2 + k$. With the given choice of $s_2 = 32(\log m + \log N)$, there is no overflow error. Thus, non-zero v_i's are never deemed to be zero vectors, and vice-versa.

B Proof of Lemma 8

By Lemma 9, it follows that $\|\tilde{V}\| = O(n^2)$. We now calculate a bound on the expected value of $\|V^{-1}\|$.

Lemma 9. $\mathbf{E}\left[\|\tilde{V}^{-1}\|\right] \leq n^2 8^n$.

Proof. Let $z_j = x_j^{1/N}$. The (i,j)th entry of the matrix V' is given by z_j^i, $0 \leq i \leq n-1$ and $1 \leq j \leq n$. Denote V'^T by W. Clearly, by definition of Frobenius norms,

$$\|V'^{-1}\|_F = \|(V'^{-1})^T\|_F = \|(V'^T)^{-1}\|_F = \|W^{-1}\|_F \ .$$

By direct calculation of the inverse of Vandermonde matrix,

$$(W^{-1})_{i,j} = \text{ coefficient of } t^i \text{ in } \frac{\prod_{\substack{s=1 \\ s\neq j}}^{n}(t - z_s)}{\prod_{\substack{s=1 \\ s\neq j}}^{n}(z_j - z_s)}$$

Consider the j^{th} column of W^{-1}. The sum of the absolute values of the entries in this column can be obtained as follows.

$$\sum_{i=1}^{n}|W_{i,j}| = \sum_{i=1}^{n} \frac{\prod_{\substack{s=1 \\ s\neq j}}^{n}(1 + z_s)}{\prod_{\substack{s=1 \\ s\neq j}}^{n}(|z_j - z_s|)}$$

We now assume that the z_i's are random, independently chosen and distinct in the domain $\{1, \ldots, N\}$. Taking expectations and using linearity of expectation,

$$\mathbf{E}\left[\sum_{i=1}^{n}|W_{i,j}|\right] = \sum_{i=1}^{n}\mathbf{E}\left[\frac{\prod_{\substack{s=1 \\ s\neq j}}^{n}(1 + z_s)}{\prod_{\substack{s=1 \\ s\neq j}}^{n}(|z_j - z_s|)}\right] \tag{12}$$

For a fixed value of x_j, the value of each of x_s, for $s \neq j$, is independent over the domain $\{1, \ldots, N\} - \{x_j\}$. Therefore,

$$\mathbf{E}\left[\frac{\prod_{\substack{s=1 \\ s\neq j}}^{n}(1 + z_s)}{\prod_{\substack{s=1 \\ s\neq j}}^{n}(|z_j - z_s|)}\right] = \prod_{\substack{s=1 \\ s\neq j}}^{n}\mathbf{E}\left[\frac{1 + z_s}{|z_j - z_s|}\right] \tag{13}$$

Since, $\frac{1}{2} \leq z_s \leq 1$, $1 + z_s \leq 2$. We therefore have,

$$\mathbf{E}\left[\frac{1 + z_s}{|z_j - z_s|}\right] \leq \mathbf{E}\left[\frac{2}{|z_j - z_s|}\right]$$

Recall that $z_j = x_j^{1/N}$ and $z_s = x_j^{1/N}$, where, $1 \leq x_j, x_s \leq N$, and $x_j \neq x_s$. For each fixed value of x_j, $|x_j - x_s|$ takes values between 1 and $N-1$ with probability at most $\frac{2}{N-1}$. Therefore,

$$\mathbf{E}\left[\frac{1}{|z_j - z_s|}\right] \leq \sum_{|x_j - x_s|=1}^{N-1} \frac{2}{N-1} \frac{1}{|z_j - z_s|} \ .$$

For a fixed value of $y = |x_j - x_s|$, the smallest value of $|z_j - z_s|$ occurs when $x_s = 1$ and $x_j = (y+1)^{1/N}$. It follows that,

$$\mathbf{E}\left[\frac{1}{|z_j - z_s|}\right] \leq \sum_{y=1}^{N-1} \frac{2}{(N-1)} \frac{1}{((y+1)^{1/N} - 1)} \ .$$

By simplifying the summation in the *RHS*, we obtain,

$$\mathbf{E}\left[\frac{1}{|z_j - z_s|}\right] \leq \frac{2}{N-1}\left(\frac{1}{2^{1/N}-1} + \int_{y=2}^{N-1} \frac{dy}{y^{1/N}-1}\right) \tag{14}$$

Note that

$$\ln 2^{1/N} = \frac{\ln 2}{N} \geq \ln\left(1 + \frac{\ln 2}{N}\right) \ .$$

Therefore,

$$2^{1/N} \geq 1 + \frac{\ln 2}{N} \text{ or, that } 2^{1/N} - 1 \geq \frac{\ln 2}{N} \ . \tag{15}$$

Substituting in (14), and replacing the variable of integration by $t = y^{1/N}$, we have,

$$\mathbf{E}\left[\frac{1}{|z_j - z_s|}\right] \leq \frac{2N}{(N-1)\ln 2} + \frac{2}{N-1}\int_{t=2^{1/N}}^{N^{1/N}} \frac{t^{N-1}dt}{t-1} \ . \tag{16}$$

Simplifying the integral in the *RHS*

$$\begin{aligned}
\int_{t=2^{1/N}}^{N^{1/N}} \frac{t^{N-1}dt}{t-1} &= \int_{t=2^{1/N}}^{N^{1/N}} (1 + t + t^2 + \ldots + t^{N-2})dt + \int_{t=2^{1/N}}^{N^{1/N}} \frac{dt}{1-t} \\
&= \left(t + \frac{t^2}{2} + \frac{t^3}{3} + \cdots + \frac{t^{N-1}}{N-1}\right)\Big|_{2^{1/N}}^{N^{1/N}} + \ln \frac{N^{1/N}-1}{2^{1/N}-1} \\
&\leq H_{N-1} + \ln \frac{1}{2^{1/N}-1} \\
&\leq H_{N-1} + 2 \ .
\end{aligned}$$

where, H_{N-1} denotes the $(N-1)^{th}$ harmonic number, namely, $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N-1}$. Substituting in (16)

$$\mathbf{E}\left[\frac{1}{|z_j - z_s|}\right] \leq \frac{2N}{(N-1)\ln 2} + \frac{2(H_{N-1}+1)}{N-1} \leq 8m \ .$$

Substituting in (13) and then in (12), we have,

$$\mathbf{E}\left[\|W^{-1}\|_F\right] = \mathbf{E}\left[\sum_{j=1}^{n}\sum_{i=1}^{n}|W_{i,j}|\right] \leq \sum_{j=1}^{n}\sum_{i=1}^{n}8^n \leq n^2 8^n \qquad \square$$

Lemma 10. $\kappa(B') \leq n^8 8^{2n} M$.

Proof. Recall that $B = \tilde{V}F\tilde{V}^T$. $\|\tilde{V}\| \leq \|\tilde{V}\|_F \leq n^2$, since, each of the entries is at most 1. The same argument holds for $\|\tilde{V}^T\|_F$. Further,

$$\|F\| \leq M \text{ and } \|F^{-1}\| \leq 1$$

Therefore,

$$\|B\| \leq \|\tilde{V}\|\|F\|\|\tilde{V}^T\| \leq n^4 M$$

By Lemma 9,

$$\|B^{-1}\| \leq \|\tilde{V}^{-1}\|\|F^{-1}\|\|(\tilde{V}^T)^{-1}\| \leq n^8 8^{2n} \ .$$

Taking products,

$$\kappa(B) = \|B\|\|B^{-1}\| \leq n^8 8^{2n} M \qquad \square$$

Proof (Of Lemma 8). The precision required for the eigenvalue computation and for computing the inverse is $O(\log \kappa(B)) = O(n + \log M) = O(n + \log M) = O(k + \log M)$. Since there are $2k + 1$ counters, the total space complexity is $(2k + 1)O(k + \log M) = O(k^2 + k \log M)$ bits. The Las Vegas method stores the counters l'_r instead of l_r. Therefore, after a total of s updates, the loss in precision is $\log s$ bits, therefore, the total number of bits required is $O(k^2 + k \log M + k \log s)$ bits. $\square$

C Review of approximate t-wise independent permutations

A family of permutations Π over F is said to be approximately t-wise independent with error parameter δ, provided, for any Turing machine M that has access to a random permutation π and makes at most t calls to π, M cannot distinguish whether π is chosen uniformly at random from Π or uniformly at random from U, with probability more than δ. Here, U is the set of all permutations over F. In other words, for any Turing machine M meeting the criterion above, and for every input I, the following condition holds.

$$\left|\mathbf{Pr}_{\pi \in_R \Pi}\{M \text{ succeeds on } I\} - \mathbf{Pr}_{\pi \in_R U}\{M \text{ succeeds on } I\}\right| \leq \delta$$

where, the notation $\pi \in_R \Pi$ indicates that π is chosen uniformly at random from Π (respectively, U). We are more interested in permutation families that are approximately t-wise

independent with *relative error* $\bar{\epsilon}$, where, $\bar{\epsilon} < 1$. That is, for any Turing machine with access to a random permutation π and that makes at most k calls to π, the following condition holds for every input I.

$$\left|\mathbf{Pr}_{\pi \in_R \Pi}\{M \text{ succeeds on } I\}\right| \in (1 \pm \bar{\epsilon})\mathbf{Pr}_{\pi \in_R U}\{M \text{ succeeds on } I\}$$

A simple calculation shows that if Π is approximately t-wise independent with relative error $\bar{\epsilon}$, then its absolute error $\delta = O(\bar{\epsilon} \cdot 2^{-mt})$.

The Fiestel permutations based approaches, such as Luby and Rackoff, Naor and Reingold [20, 25] have errors of the form of $\delta = \frac{t^2}{2^{m/2}}$, that are considerably large and inadequate. The 3-bit mixed permutations of Gowers [14] with improvements due to Hoory, Magen, Myers and Rackoff [16] require space and time $O((m^3t^2 + m^2t\log\frac{1}{\bar{\epsilon}})\log m)$ bits and m-bit word operations to evaluate $\pi(x)$, respectively. The construction of Kaplan, Naor and Reingold [19] results in $\bar{\epsilon}$-approximate t-wise independent permutation family requiring $O(mt + \log\frac{1}{\bar{\epsilon}})$ bits. However, the time complexity of the Kaplan, Naor, Reingold construction is a very high-degree of polynomial in m and t (also discussed by the authors in [19] Section 6.1).

In data stream processing, π has to be applied to each arriving record, and since, records arrive at a rapid rate, $\pi(\cdot)$ must be very efficiently computable, which, unfortunately, is not the case with Kaplan, Naor and Reingold's construction. We therefore use the construction of Hoory, Magen, Myers and Rackoff since its time-complexity is relatively better, but remark that it is desirable to obtain constructions of approximate t-wise independent permutation families that are more suited for high-speed data stream processing.