DIMACS Technical Report 2002-35
Oringinally Submitted July 2002, Revised June 2003

Estimating Dominance Norms of Multiple Data Streams

by

Graham Cormode[1,2]   S. Muthukrishnan[3,4]

[1]DIMACS Postdoctoral Fellow
[2]Center for Discrete Mathematics and Computer Science, Rutgers University, New Jersey USA, `graham@dimacs.rutgers.edu`
[3]Division of Computer Science, Rutgers University, New Jersey USA, `muthu@cs.rutgers.edu` and AT&T Research
[4]DIMACS Permanent Member

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, NEC Research Institute and Telcordia Technologies (formerly Bellcore).

DIMACS was founded as an NSF Science and Technology Center, and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

There is much focus in the database and algorithms communities on designing tools to manage and mine data streams. Typically, data streams consist of multiple signals. Formally, a stream of multiple signals is $(i, a_{i,j})$ where i's correspond to the domain, j's index the different signals and $a_{i,j} \geq 0$ give the value of the jth signal at point i. We study the problem of finding norms that are cumulative of the multiple signals in the data stream.

For example, consider the max-dominance norm, defined as $\sum_i \max_j\{a_{i,j}\}$. It may be thought as estimating the norm of the "upper envelope" of the multiple signals, or alternatively, as estimating the norm of the "marginal" distribution of tabular data streams. It is used in applications to estimate the "worst case influence" of multiple processes, for example in IP traffic analysis, electrical grid monitoring and financial domain. In addition, it is a natural measure, generalizing the union of data streams or counting distinct elements in data streams.

We present the first known data stream algorithms for estimating max-dominance of multiple signals. In particular, we use workspace and time-per-item that are both sublinear (in fact, poly-logarithmic) in the input size. In contrast we show that other notions of dominance on streams a, b — min-dominance ($\sum_i \min_j\{a_{i,j}\}$), count-dominance ($|\{i|a_i > b_i\}|$) or relative-dominance ($\sum_i a_i/\max\{1, b_i\}$) — are all impossible to estimate accurately with sublinear space.

1 Introduction

Data streams are emerging as a powerful, new data source. Data streams comprise data generated rapidly over time in massive amounts; each data item must be processed quickly as it is generated. Often, only a limited amount of high speed memory and processing power is available. Data streams arise in monitoring telecommunication networks, sensor observations, financial transactions, etc. A significant scenario — and our motivating application — arises in IP networking where Internet Service Providers (ISPs) monitor (a) logs of total number of bytes or packets sent per minute per link connecting the routers in the network, or (b) logs of IP "flow" which are roughly distinct IP sessions characterized by source and destination IP addresses, source and destination port numbers etc. on each link, or at a higher level (c) logs of web clicks and so on. Typically, the logs are monitored in near-real time for simple indicators of "actionable" events, such as anomalies, large concurrence of faults, "hot spots", and surges, as part of the standard operations of ISPs. Systems in such applications need flexible methods to define, monitor and mine such indicators in data streams.

The starting point of our investigation here is the observation that data streams are often not individual signals, but they comprise multiple signals presented in an interspersed and distributed manner. For example, web click streams may be arbitrary ordering of clicks by different customers at different web servers of a server farm; financial events may be stock activity from multiple customers on stocks from many different sectors and indices; and IP traffic logs may be logs at management stations of the cumulative traffic at different time periods from multiple router links. Even at a single router, there are several — up to a hundred or so — interfaces, each of which has multiple logs of traffic data. Our focus here from a conceptual point of view is on suitable norms to measure and monitor about the set of all distributions we see in the cumulative data stream.

Previous work on norm estimation and related problems on data streams has been extensive, but primarily focused on *individual* distributions. This includes estimating frequency moments [AMS96], wavelet coefficients [GKMS01], histograms [GKS01, TIGK02], and other indicators. For the case of multiple distributions, prior work has typically focused on processing each distribution *individually* so that multiple distributions can be compared based on estimating pairwise distances such as L_p norms [FKSV99, FS00, Ind00]. These L_p norms are linear so that the per-distribution processing methods can be used to index, cluster multiple distributions or do proximity searches; however, all such methods involve storing space proportional to the number of distinct distributions in the data stream. As such, they do not provide a mechanism to directly understand trends in multiple distributions.

Our motivating scenario is one of a rather large number of distributions as in the IP network application above. In particular, we initiate the study of norms for cumulative trends in presence of multiple distributions. For some norms of this type (in particular, the max-dominance norm to be defined soon), we present efficient algorithms in the data stream model; interestingly, these use space independent of the number of distributions in the signal. For a few other norms, we show hardness results. In what follows, we provide details on the data stream model, on dominance norms and our results.

1.1 Data Stream Model

As the idea of computations on the data stream has risen in popularity, a number of different formalizations have been provided by different authors, depending on the nature of the

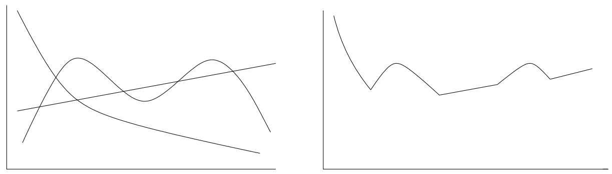

Figure 1: A mixture of distributions (left), and their upper envelope (right)

problem they are describing. We give a brief summary of the notation we will be using to handle data streams. In order to formalize our discussion here, we let the data stream be series of items $(i, a_{i,j})$ presented in some *arbitrary order*; i's correspond to the domain of the distributions (assumed to be identical without loss of generality), j's to the different distributions and $a_{i,j}$ is the value of the distribution j at location i (we will assume $0 \leq a_{i,j} \leq M$ for discussions here).

Note that there is no relation between the order of arrival and the parameter i, which indexes the domain, or j, which indexes the signals. For convenience of notation, we use the index j to indicate that $a_{i,j}$ is from signal j, or is the jth tuple with index i. However, j is not generally made explicit in the stream and we assume it is not available for the processing algorithm to use. We use n_i to denote the number of tuples for index i seen so far, and $n = \sum_i n_i$ is the total number of tuples seen in the data stream. There are three parameters of the algorithm that are of interest to us: the amount of space used; the time used to process each item that arrives; and the time taken to produce the approximation of the quantity of interest. For an algorithm that works in the data stream to be of interest, the working space and per item processing time must both be sublinear in n and M, and ideally poly-logarithmic in the these quantities.

1.2 Dominance Norms and their Relevance

We study norms that are cumulative over the multiple distributions in the data stream. We particularly focus on the *max-dominance* defined as

$$\textstyle\sum_i \max_j\{a_{i,j}\}$$

Intuitively, this corresponds to computing the L_1 norm of the upper envelope of the distributions, illustrated schematically in Figure 1. We can alternately represent this as a problem of finding information about the *marginals* of multidimensional input tuples: let $a_{i,j}$ define a matrix of (two dimensional) input entries. Then the marginal value of each row is some function of that row — in this case the maximum of the entries — and the dominance norm of the matrix is the sum of these marginal values. We later consider the cases where this dominance is based on marginals found by minimum, median, and other similar functions.

Computing the max-dominance norm of multiple data distributions is interesting for many important reasons described below. First, applications abound where this measure is suitable for estimating the "worst case influence" under multiple distributions. For example, in the IP network scenario, i's correspond to source IP addresses and $a_{i,j}$ corresponds to the number of packets sent by IP address i in the jth transmission. Here the max-dominance measures the maximum possible utilization of the network if the transmissions from different source IP addresses were coordinated. A similar analysis of network capacity using max-dominance is relevant in the electrical grid [Ele] and in other instances with IP networks, such as using SNMP [LSC]. The concept of max-dominance occurs in financial applications, where the maximum dollar index (MDI) for securities class action filings characterize the intensity of litigation activity through time [Sec]. In addition to finding specific applications, max-dominance norm has intrinsic conceptual interest in the following two ways.

- If $a_{i,j}$'s were all 0 or 1, then this norm reduces to calculating the size of the union of the multiple sets. Therefore max-dominance norm is a generalization of the standard union operation.

- Max-dominance can be viewed as a generalization of the problem of counting the number of distinct elements i that occur within a stream. The two norms again coincide when each $a_{i,j}$ takes on binary values.

We denote the *max-dominance* of such a stream a as $\mathrm{dom}_{\max}(a) = \sum_i \max_{1 \le l \le n_i}\{a_{i,l}\}$. Equivalently, we define the i'th entry of an implicit state vector as $\max_{1 \le l \le n_i}\{a_{i,l}\}$, and the $\mathrm{dom}_{\max}$ function is the L_1 norm of this vector. Note that it is a Norm, as $\mathrm{dom}_{\max}(\mathbf{0}) = 0$, $\mathrm{dom}_{\max}(ka) = k\,\mathrm{dom}_{\max}(a)$ and $\mathrm{dom}_{\max}(a+b) \le \mathrm{dom}_{\max}(a) + \mathrm{dom}_{\max}(b)$. In addition to the example of max-dominance, there are other dominance norms with multiple distributions. Closely related to max-dominance norms is *min-dominance* ($\mathrm{dom}_{\min}$) and *median-dominance* — $\sum_i \min_j\{|a_{i,j}|\}$ and $\sum_i \mathrm{median}_j\{|a_{i,j}|\}$ — or more generally $\sum_i \mathrm{quantiles}_j\{|a_{i,j}|\}$). Generalizing these measures on various orderings (not just quantiles) of values are relative measures of dominance: *Relative count dominance* is based on counting[1] the number of places where one distribution dominates another (or others, more generally), $|\{i | a_i > b_i\}|$ for two given data distributions a and b, and *relative sum dominance* which is $\sum_i\{\frac{a_i}{\max\{1,b_i\}}\}$.[2] All of these dominances are very natural for collating information from two or more signals in the data stream.

1.3 Our Results

Our contributions are as follows.

1. We initiate the study of dominance norms as indicators for collating information from multiple signals in data streams.

2. We present streaming algorithms for maintaining the max-dominance of multiple data streams. Our algorithms estimate max-dominance to $1+\epsilon$ approximation with probability at least $1-\delta$. We show an algorithm that uses $O(\frac{\log M \log\log^2 M}{\epsilon^3})$ by reducing the problem to multiple instances of the problem of estimating distinct items on data streams. However, the main part of our technical contribution is an improved algorithm that uses only $O(\frac{\log M}{\epsilon^2})$ space. In both cases, the running time as well as time to compute the norm is also similarly polylogarithmic. This is the bulk of our technical work. No such sublinear space and time result was known for estimating any dominance norms in the data stream model.

[1] The relative counting dominance norm has previously arisen in string matching context [AF91]. However, the focus there was not on streaming.

[2] The problem of estimating the relative norm $\sum_i\{\frac{a_i}{\max\{1,b_i\}}\}$ has been proposed as an open problem in [CCFC02] and in a talk by Monika Henzinger at the AMS-SIAM Joint meeting in January 2002.

3. We show that, in contrast, all other closely related dominance norms — min-dominance, relative count dominance and relative sum dominance — need linear space to be even probabilistically approximated in the streaming model. These use reductions from other problems known to be hard in the streaming and communication complexity models.

The rest of the paper is organized as follows. Next, we discuss related work in the data stream model on similar problems. In Section 2, we show how a solution can be made by using existing solutions to the problem of computing the number of distinct elements in the stream. We give our improved solution based on the use of stable distributions, and necessary technical results, in Section 3. Section 4 presents our hardness results that show dominance norms other than max-dominance to be hard to (even) approximate using sublinear space. Concluding Remarks are in Section 5.

1.4 Related Work

Surprisingly, almost no data stream algorithms are known for estimating any of the dominance norms, although recent work has begun to investigate the problems involved in analyzing and comparing multiple data streams [YSJ+00]. There, the problem is to predict missing values, or determine variations from expected values, in multiple evolving streams. Much of the recent flurry of results in data streams has focused on using various L_p norms for *individual* distributions to compare and collate information from different data streams, for example, $(\sum_i(\sum_j a_{i,j})^p)^{1/p}$ for $0 < p \leq 2$ [AMS96, FKSV99, Ind00, CIKM02] and related notions such as Hamming norms $\sum_i((\sum_j a_{i,j}) \neq 0)$ [CDIM02]. While these norms are suitable for capturing comparative trends in multiple data streams, they are not applicable for computing the various dominance norms (max, min, count or relative). Most related to the methods here is our work in [CDIM02], where we used Stable Distributions with small parameter. We extend this work by applying it to a new scenario, that of dominance norms. Here we need to derive new properties: the behavior of these distributions as the parameter approaches zero (Sections 3.4—3.6), how range sums of variables can be computed efficiently (Section 3.6), and so on.

Also relevant is work on computing the number of distinct values within a stream, which has been the subject of much study [FM83, Gib01, GT01, CDIM02, BYJK+02, BYKS02]. In [GT01], the authors mention that their algorithm can be applied to the problem of computing $\sum_i \max a_i, b_i$, which is a restricted version of our notion of dominance norm. Applying their algorithm directly yields a time cost of $\Omega(M)$ to process each item, which is prohibitive for large M (it is exponential in the input size). Other approaches which are used in ensemble as indicators when observing data streams include monitoring those items that occur very frequently (the "heavy hitters" of [FGL+00, EV01]), and those that occur very rarely [DM02]. We mention only work related to our current interest in computing dominance norms of streams. For a more general overview of issues and algorithms in processing data streams, see the survey [Mut03].

2 Max-Dominance Norms using Distinct Element Counting

Let us first elaborate on the challenge in computing dominance norms of multiple data streams by focusing on the max-dominance norm. If we had space proportional to the range of values i then for each i, we can store $\max_j\{|a_{i,j}|\}$ for all the $a_{i,j}$'s seen thus far, and incrementally maintain $\sum_i \max_j\{|a_{i,j}|\}$. However, in our motivating scenarios, the inputs arrive as streams, and we need to work with sublinear space (say poly-logarithmic in the range of the is, the total number of tuples, n, and in M, the range of the $a_{i,j}$s) and take very little time (also poly-logarithmic in these quantities) for processing per item as well as for estimating the dominance norm. In this data stream model, algorithms for computing max-dominance norms are no longer obvious.

A simple approach involves using distinct elements counting. Recall that our problem can be viewed as a generalization of the problem of counting the number of distinct elements that occur within a stream. The two norms coincide when each $a_{i,j}$ takes on values either 0 or 1, and it turns out that techniques for counting the number of distinct elements give insight into producing solutions for Max-Dominance. We develop algorithms that reduce the max-dominance estimation problem to multiple instances of counting the number of distinct values in a stream, whose complexity depends on the complexity of these algorithms. Thus, a distinct element counting algorithm can be used as a black-box. In this case, any method which can approximate this value sufficiently accurately will suffice, such as the Distinct Sampling technique [Gib01, GT01], the methods described in [BYJK+02] or using stable distributions as in [CDIM02] for counting distinct elements. We do not discuss any particulars of these referenced algorithms in this abstract.

Theorem 2.1 *By maintaining $\frac{\log M}{\epsilon}$ independent copies of a method for counting distinct values in the stream, we can compute a $1 \pm \epsilon$ approximation to the dominance norm with probability $1-\delta$. The per-element processing time is that needed to insert an element into $O(\log a_{i,j}/\epsilon)$ of the distinct elements algorithms.*

Proof: We require access to $K = \lceil \log M/\log(1+\epsilon)\rceil + 1 = \log M/\epsilon + O(1)$ different instantiations of distinct elements algorithms. We shall refer to these as $D_0 \ldots D_k \ldots D_K$. On receiving a tuple $(i, a_{i,j})$, we compute the 'level', l, of this item as $l = \lceil \frac{\log a_{i,j}}{\log 1+\epsilon}\rceil$. We then insert the identifier i into certain of the distinct element algorithms: those D_k where $0 \le k \le l$. Let D_k^{out} indicate the approximation of the number of distinct elements of D_k. The approximation of the dominance norm of the sequence is given by:

$$\hat{d}(a) = D_0^{out} + \sum_{j=1}^{K}(\lceil(1+\epsilon)^j\rceil - \lceil(1+\epsilon)^{j-1}\rceil)D_j^{out}$$

We consider the effect of any individual i, which is represented in the stream by multiple values $a_{i,j}$. By the effect of the distinct elements algorithms, the contribution is 1 at each level up to $\lceil \log(\max_j a_{i,j})/\log(1+\epsilon)\rceil$. The effect of this on the scaled sum is then between $\max_j a_{i,j}$ and $(1+\epsilon)\max_j a_{i,j}$ if each distinct element algorithms give the exact answer. This procedure is illustrated graphically in Figure 2. Since these are actually approximate, then we find a result between $(1-\epsilon)\max_j a_{i,j}$ and $(1+\epsilon)^2 \max_j a_{i,j}$. Summing this over all i, we get

$(1-\epsilon)\,\mathrm{dom}_{\max}(a) \le \hat{d}(a) \le (1+\epsilon)^2\,\mathrm{dom}_{\max}(a)$ ∎

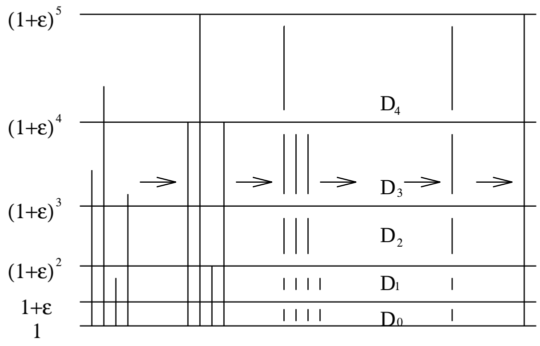

Figure 2: Each item is rounded to the next value of $(1+\epsilon)^l$. We send every $k < l$ to a distinct elements counter D_k. The output of these counters is scaled appropriately, to get back the maximum value seen (approximate to $1+\epsilon$)

Corollary 1 *There is an algorithm for computing Dominance norms which outputs a* $(1+\epsilon)$ *approximation with probability* $1-\delta$ *which uses* $\tilde{O}(1/\epsilon \log M(1/\epsilon^2 + \log M) \log \frac{\log M}{\delta\epsilon})$ *space, and amortized time* $\tilde{O}(1/\epsilon \log^2 M \log \frac{\log M}{\delta\epsilon})$ *per item.*

This follows by adopting the third method described in [BYJK+02], which is the most space efficient method for finding the number of distinct elements in a stream that is in the literature. The space required for each D is $O((1/\epsilon^2 + \log M)\log(1/\epsilon)\log\log(M)\log 1/\delta)$. Updates take amortized time $O(\log M + \log 1/\epsilon)$. In order to have probability $1-\delta$ of every count being accurate within the desired bounds, we have to increase the accuracy of each individual test, replacing δ with $\epsilon\delta/\log M$. Putting this together with the above theorem gets the desired result. Clearly, with better methods for computing the number of distinct elements, better results could be obtained. ∎

A variation that was considered was the List-Efficient method given in [BYKS02]. This allows the simulation of multiple levels to be done by the counting algorithm itself. However, the properties of this method are stronger than we need, and consequentially the time complexity of this approach, $\Omega(\frac{1}{\epsilon^5}\log^5 M \log 1/\delta)$, is significantly higher than any of the methods we have described here, so we do not follow this direction further.

3 Max-Dominance Norm via Stable Distributions

We next present a second method to compute the max-dominance of a stream which has an improved space requirement, and offers some additional functionality. It relies on the use of random variables whose combination is precisely the max-dominance that we require. These can be simulated using stable probability distributions for extremely small stability parameter $\alpha > 0$. We take inspiration from the pioneering work of [Ind00] that introduced the use of stable distributions to data stream algorithms; however, the application of stable distributions with tiny α to this problem is novel and extends previous work. A technical crux of the work is to perform a detailed analysis of stable distributions with tiny α which may be of independent interest. The remainder of our work on max-dominance in this approach attends to important practical details: how to generate these random variables in small space, and how to speed up the algorithm so we do not take time $O(a_{i,j})$ per item which is prohibitive. We provide these details which is the bulk of our technical work.

3.1 Stable Distributions

Indyk pioneered the use of Stable Distributions in data streams and since then have received a great deal of attention [Ind00, CIKM02, CDIM02, GGI+02]. They have been used to generalize the Johnson-Lindenstrauss lemma [JL84] (which approximates the Euclidean distance) to all L_p distances with $0 < p \leq 2$ [Ind00]. Throughout our discussion of distributions, we shall use $\sim$ for the equivalence relation meaning "is equivalent in distribution to".

A stable distribution is defined by four parameters. These are (i) the stability index, $0 < \alpha \leq 2$; (ii) the skewness parameter, $-1 \leq \beta \leq 1$; (iii) scale parameter, $\gamma > 0$; and (iv) location parameter, δ. Throughout we shall deal with a canonical representation of stable distributions, where $\gamma = 1$ and $\delta = 0$. Therefore, we consider stable distributions $S(\alpha, \beta)$ so that, given α and β the distribution is uniquely defined by these parameters. We write $X \sim S(\alpha, \beta)$ to denote the random variable X is distributed as a stable distribution with parameters α and β. When $\beta = 0$, as we will often find, then the distribution is symmetric about the mean, and is called *strictly* stable.

Definition 3.1 *The strictly stable distribution $S(\alpha, 0)$ is defined by the property that given independent variables distributed stable:*

$$X \sim S(\alpha, 0), Y \sim S(\alpha, 0), Z \sim S(\alpha, 0) \Rightarrow aX + bY \sim cZ, a^\alpha + b^\alpha = c^\alpha$$

That is, if X and Y are distributed with stability parameter α, then any linear combination of them is also distributed as a stable distribution with the same stability parameter α. The result is scaled by the scalar c where $c = |a^\alpha + b^\alpha|^{1/\alpha}$. The definition uniquely defines a distribution, up to scaling and shifting. By centering the distribution on zero and fixing a scale, we can talk about *the* strictly stable distribution with index α. From the definition it follows that

$$X_1 \ldots X_n \sim S(\alpha, 0);\ \boldsymbol{a} = (a_1, \ldots, a_n);$$

$$||\boldsymbol{a}||_\alpha = (\textstyle\sum_i |a_i|^\alpha)^{1/\alpha} \Rightarrow \sum_i a_i X_i \sim ||\boldsymbol{a}||_\alpha S(\alpha, 0)$$

3.2 Our Result

Recall that we wish to compute the sum of the maximum values seen in the stream. That is, we want to find $\mathrm{dom}_{\max}(a) = \sum_i \max\{a_{i,1}, a_{i,2}, \ldots a_{i,n_i}\}$.
If each index i appears at most once, then this is trivial to compute exactly, since it is just $\sum_i a_{i,1}$. Similarly, if each index appears at most c times, then the quantity $\sum_{i,j} a_{i,j}$ is a c-approximation to the max-dominance of the stream. However, in general we have no such guarantee nor any reason to expect one, and so we must assume that each index may be represented many times over by many different signals. We will show how the max-dominance can be found approximately by using values drawn from stable distributions. This allows us to state our main theorem:

Theorem 3.1 *It is possible to compute an approximation to $\sum_i \max_{1 \leq j \leq n_i}\{a_{i,j}\}$ in the streaming model that is correct within a factor of $(1+\epsilon)$ with probability $1-\delta$ using space $O(1/\epsilon^2(\log M + \epsilon^{-1}\log n \log\log n)\log 1/\delta)$ and taking time $O(1/\epsilon^4 \log a_{i,j} \log n \log 1/\delta)$ per item.*

3.3 Idealized Algorithm

We first give an outline algorithm, then go on to show how this algorithm can be applied in practice on the stream with small memory and time requirements. We imagine that we have access to a special indicator distribution X. This has the (impossible) property that for any positive integer c (that is, $c > 0$) then $E(cX) = 1$ and bounded variance. From this it is possible to derive a solution problem of finding the max-dominance of a stream of values. We maintain a scalar z, initially zero. We create a set of $x_{i,k}$, each drawn from iid distributions $X_{i,k} \sim X$. For every $a_{i,j}$ in the input streams we update z as follows:

$$z \leftarrow z + \sum_{k=1}^{a_{i,j}} x_{i,k}$$

This maintains the property that the expectation of z is $\sum_i \max_j\{a_{i,j}\}$, as required. This is a consequence of the "impossible" property of $X_{i,k}$ that it contributes only 1 to the expectation of z no matter how many times it is added. For example, suppose our stream consists of $\{(i = 1, a_{1,1} = 2), (3, 3), (3, 5)\}$. Then z is distributed as $X_{1,1} + X_{1,2} + 2X_{3,1} + 2X_{3,2} + 2X_{3,3} + X_{3,4} + X_{3,5}$. The expected value of z is then the number of different terms, 7, which is the max dominance that we require (2+5). The required accuracy can be achieved by in parallel keeping several different values of z based on independent drawings of values for $x_{i,k}$. There are a number of hurdles to overcome in order to turn this idea into a practical solution.

1. **How to choose the distributions $X_{i,k}$?** We shall see how appropriate use of stable distributions can achieve a good approximation to these indicator variables.

2. **How to reduce space requirements?** The above algorithm requires repeated access to $x_{i,k}$ for many values of i and k. We need to be able to provide this access without explicitly storing every $x_{i,k}$ that is used. We also need to show that the required accuracy can be achieved by carrying out only a small number of independent repetitions in parallel.

3. **How to compute efficiently?** We require fast per item processing, that is polylogarithmic in the size of the stream and the size of the items in the stream. But the algorithm above requires adding $a_{i,j}$ different values to a counter in each step: time linear in the size of the data item (that is, exponential in the size of its binary representation). We show how to compute the necessary range sums efficiently while ensuring that the memory usage remains limited.

For each hurdle, we shall how it can be cleared whilst working within the streaming model.

3.4 Our Algorithm

We will use stable distributions with small stability parameter α in order to *approximate* the indicator variable $X_{i,k}$. Stable distributions can be used to approximate the number of non-zero values in a vector [CDIM02]. For each index i, we can consider a vector $\boldsymbol{a}(i)$ defined by the tuples for that index i along, so that $\boldsymbol{a}(i)_k = |\{j|a_{i,j} \geq k\}|$. Then the number of non-zero entries of $\boldsymbol{a}(i) = \max_j a_{i,j}$. We shall write $\boldsymbol{a}$ for the vector formed by concatenated all such vectors for different i. This is an alternate representation of the stream, a. To approximate the max-dominance, we will maintain a sketch vector $\boldsymbol{z}(a)$ which summarizes the stream a.

Definition 3.2 *The* sketch vector $\boldsymbol{z}(a)$ *has a number of entries* m $(= O(1/\epsilon^2 \log 1/\delta))$*. We make use of a number of values* $x_{i,k,l}$*, each of which is drawn independently from* $S(\alpha, 0)$*, for* $\alpha = \epsilon/\log n$*. Initially* $\boldsymbol{z}$ *is set to zero in every dimension.*

Invariant. *We maintain the property for each* l *that* $\boldsymbol{z}_l = \sum_{i,j} \sum_{k=1}^{a_{i,j}} x_{i,k,l}$

Update Procedure. *On receiving each pair* $(i, a_{i,j})$ *in the stream, we maintain the invariant by updating* $\boldsymbol{z}$ *as follows:* $\forall 1 \leq l \leq m.\ \boldsymbol{z}_l \leftarrow \boldsymbol{z}_l + \sum_{k=1}^{a_{i,j}} x_{i,k,l}$

Output. *Our approximation of the max-dominance norm is* $\ln 2(\mathrm{median}_l\, |\boldsymbol{z}_l|)^\alpha$

At any point, it is possible to extract from the sketch $\boldsymbol{z}(a)$ a good approximation of the sum of the maximum values.

Theorem 3.2 $(1-\epsilon)\,\mathrm{dom}_{\max}(a) \leq \ln 2\ (\mathrm{median}_l\, |\boldsymbol{z}(a)_l|)^\alpha \leq (1+\epsilon)^2\,\mathrm{dom}_{\max}(a)$

Proof: From the defining property of stable distributions (Definition 3.1), we know by construction that each entry of $\boldsymbol{z}$ is drawn from the distribution $||\boldsymbol{a}||_\alpha S(\alpha, 0)$. We know that we will add any $x_{i,k,l}$ to $\boldsymbol{z}_l$ at most once for each tuple in the stream, so we have an upper bound $U = n$ on each entry of $\boldsymbol{a}$. A simple observation is that for small enough α and an integer valued vector then the norm $||\boldsymbol{a}||_\alpha^\alpha$ (L_α norm raised to the power α, which is just $\sum_i \boldsymbol{a}_i^\alpha$) approximates the number of non-zero entries in the vector. Formally, if we set an upper bound U so that $\forall i. |\boldsymbol{a}_i| \leq U$ and fix $0 < \alpha \leq \epsilon/\log_2 U$ then

$$\begin{aligned} |\{i|\boldsymbol{a}_i \neq 0\}| &= \sum_{\boldsymbol{a}_i \neq 0} 1^\alpha \leq \sum_{\boldsymbol{a}_i \neq 0} |\boldsymbol{a}_i|^\alpha = ||\boldsymbol{a}||_\alpha^\alpha \\ &\leq \sum_{\boldsymbol{a}_i \neq 0} U^\alpha \leq \exp(\epsilon \ln 2)|\{i|\boldsymbol{a}_i \neq 0\}| \\ &\leq (1+\epsilon)|\{i|\boldsymbol{a}_i \neq 0\}| \end{aligned}$$

Using this, we choose α to be $\epsilon/\log_2 n$ since each value i appears at most n times within the stream, so $U = n$. This guarantees

$$\mathrm{dom}_{\max}(a) \leq ||\boldsymbol{a}||_\alpha^\alpha \leq (1+\epsilon)\,\mathrm{dom}_{\max}(a)$$

Lemma 3.1 *If* $X \sim S(\alpha, \beta)$ *then*

$$\lim_{\alpha \to 0^+} \mathrm{median}(|cX|^\alpha) = |c|^\alpha\, \mathrm{median}(|X|^\alpha) = \tfrac{|c|^\alpha}{\ln 2}$$

Proof: Let E be distributed with the exponential distribution with mean one. Then $\lim_{\alpha \to 0^+} |S(\alpha, \beta)|^\alpha = E^{-1}$ [Cre75]. The density of E^{-1} is $f(x) = x^{-2}\exp(-1/x),\ x > 0$ and the cumulative density is

$$F(x) = \int_0^x f(x)dx = \exp(-1/x)$$

so $\mathrm{median}(E^{-1}) = F^{-1}(1/2) = 1/\ln 2$ ∎

Consequently $\forall k. |\boldsymbol{z}_k|^\alpha \sim ||\boldsymbol{a}||_\alpha^\alpha |X|^\alpha$ and $\mathrm{median}|\, ||\boldsymbol{a}||_\alpha X|^\alpha \to ||\boldsymbol{a}||_\alpha^\alpha/\ln 2$. It is noteworthy that analysis of the scaling constant $\ln 2$ is the sharpest yet known for the behavior of stable distribution process with small α. Previously, several results presented estimates for the necessary scaling constant with only experimental evidence [CDIM02].

We next make use of a standard sampling result:

Lemma 3.2 *Let X be a distribution with cumulative density function $F(x)$. If derivative of the inverse of $F(X)$ is bounded by a constant around the median then the median of $O(1/\epsilon^2 \log 1/\delta)$ samples from X is within a factor of $1 \pm \epsilon$ of* $\mathrm{median}(X)$ *with probability $1-\delta$.*

The derivative of the inverse density is indeed bounded at the median in the limit, since $F^{-1}(r) = -1/\ln r$, and $(F^{-1})'(\frac{1}{2}) < 5$. Hence for a large enough constant c, by taking a vector $\boldsymbol{z}$ with $m = c/\epsilon^2 \log 1/\delta$ entries, each based on an independent repetition of the above procedure, then we can approximate the desired quantity and so $(1-\epsilon)||\boldsymbol{a}||_\alpha^\alpha \leq \ln 2 \, \mathrm{median}_k \, |\boldsymbol{z}_k|^\alpha \leq (1+\epsilon)||\boldsymbol{a}||_\alpha^\alpha$ with probability $1-\delta$ by this Lemma. ∎

Thus to find our approximation of the sum of the maximum values, we maintain the vector $\boldsymbol{z}$ as the dot product of the underlying vector $\boldsymbol{a}$ with the values drawn from stable distributions, $x_{i,k,l}$. When we take the absolute value of each entry of $\boldsymbol{z}$ and find their median, the result raised to the power α and scaled by the factor of $\ln 2$ is the approximation of $\mathrm{dom}_{\max}(a)$.

3.5 Space Requirement

For the algorithm to be applicable in the streaming model, we need to ensure that the space requirements are minimal, and certainly sublinear in the size of the stream. Therefore, we cannot explicitly keep all the values we draw from stable distributions, yet we require the values to be the same each time the same entry is requested at different points in the algorithm. This problem can be solved by using pseudo-random generators: we do not store any $x_{i,k,l}$ explicitly, instead we create it as a pseudo-random function of k, i, l and a small number of stored random bits whenever it is needed. We need a different set of random bits in order to generate each of the m instantiations of the procedure. We therefore need only consider the space required to store the random bits, and to hold the vector $\boldsymbol{z}$. It is known that although there is no closed form for stable distributions for general α, it is possible to draw values from such distributions for arbitrary α by using a transform from two independent uniform random variables.

Lemma 3.3 (Equation (2.3) of [CMS76]) *Let U be a uniform random variable on $[0,1]$ and Θ uniform on $[\frac{-\pi}{2}, \frac{\pi}{2}]$. Then*

$$S(\alpha, 0) \sim \frac{\sin \alpha\Theta}{(\cos\Theta)^{1/\alpha}} \left(\frac{\cos(1-\alpha)\Theta}{-\ln U} \right)^{\frac{1-\alpha}{\alpha}}$$

We also make use of two other results on random variables from the literature (see for example [UZ99]), with which we will prove the space requirements for the algorithm.

Lemma 3.4 *(i) $Y \sim S(\alpha,1), Z \sim S(\alpha,1) \Rightarrow 2^{-1/\alpha}(Y-Z) \sim S(\alpha,0)$*
(ii) In the limit, the density function $f(x)$ obeys $X \sim S(\alpha,1), \alpha \to 0^+ \Rightarrow f(x) = O(\alpha \exp(-x^{-\alpha})x^{-\alpha-1})$, $x > 0$

Lemma 3.5 *The space requirement of this algorithm is $O(1/\epsilon^2(\log M + \frac{1}{\epsilon}\log n)\log(1/\delta))$ bits.*

Proof: For each repetition of the procedure, we require $O(\log n)$ random bits to instantiate the pseudo-random generators, as per [Nis92, Ind00]. We also need to consider the space used to represent each entry of $\boldsymbol{z}$. We analyze the process at each step of the algorithm: a value x is drawn (pseudo-randomly) from $S(\alpha, 0)$, and added to an entry in $\boldsymbol{z}$. The number of bits needed to represent this quantity is $\log_2 |x|$. Since the cumulative distribution of the limit from Lemma 3.4 (ii) is

$$F_{\beta=1}(x) = \int_0^x \alpha \exp(-x^{-\alpha}) x^{-\alpha-1} dx = \exp(-x^{-\alpha})$$

then $F^{-1}_{\beta=1}(r) = (\ln r^{-1})^{-1/\alpha} \quad 0 \leq r \leq 1$
So $|x| = O(F^{-1}_{\beta=0}(r)) = O(2^{-1/\alpha}(\ln r^{-1})^{-1/\alpha})$ by Lemma 3.4 (i). Therefore $\log_2 |x| = O(1/\alpha \log \ln r)$. The dependence on α is $O(\alpha^{-1})$, which was set in Theorem 3.2 as $\alpha \leq \epsilon / \log n$. The value of r requires a poly-log number of bits to represent, so representing x requires $O(\frac{1}{\epsilon} \log n \log \log n)$ bits. Each entry of $\boldsymbol{z}$ is formed by summing many such variables. The total number of summations is bounded by Mn. So the total space to represent each entry of $\boldsymbol{z}$ is

$$\log \boldsymbol{z}_k = \tilde{O}(\log Mnx) = \tilde{O}(\log M + \log n + \tfrac{1}{\epsilon} \log n)$$

The total space required for all $O(1/\epsilon^2 \log 1/\delta)$ entries of $\boldsymbol{z}$ is $O(1/\epsilon^3 \log 1/\delta \log n \log \log n)$ if we assume M is bounded by a polynomial in n. ∎

3.6 Per Item Processing Time

For each item, we must compute several sums of variables drawn from stable distributions. Directly doing this will take time proportional to $a_{i,j}$. We could precompute sums of the necessary variables, but we wish to avoid explicitly storing any values of variables to ensure that the space requirement remains sublinear. However, the defining property of stable distributions is that the sum of any number of variables is distributed as a stable distribution.

Lemma 3.6 *The sum $\sum_{k=1}^{a_{i,j}} x_{i,k}$ can be approximated up to a factor of 2 in $\log a_{i,j}$ steps.*

Proof: Suppose we replaced each $a_{i,j}$ with the smallest power of 2 that is at least as big, that is, with $2^{\lceil \log_2 a_{i,j} \rceil}$. If we find $\mathrm{dom}_{\max}$ of this modified stream, then this will be a 2-approximation of the true answer. Applying this rounding procedure, we can find the sum of variables drawn from stable distributions by the identity:

$$\sum_{k=1}^{2^s} x_{i,k} = x_{i,1} + x_{i,2} + \sum_{k=3}^{4} x_{i,k} + \ldots + \sum_{k=2^s+1}^{2^s} x_{i,k} = x_{i,1} + \sum_{a=0}^{s} \sum_{k=2^a+1}^{2^{a+1}} x_{i,k}$$

Each term in this sum is a sum of 2^a variables with stable distribution. From Definition 3.1, we know that $\sum_{k=3}^{4} x_{i,k} \sim S(\alpha, 0) + S(\alpha, 0) \sim 2^{1/\alpha} S(\alpha, 0)$. Generalizing,

$$\sum_{k=2^a+1}^{2^{a+1}} x_{i,k} \sim 2^{a/\alpha} S(\alpha, 0)$$

Hence to find the desired sum, we only need to find $\log_2 n$ values drawn from $S(\alpha, 0)$, and compute their scaled sum. By using the pseudo-random method in Lemma 3.5, we guarantee that the values found will be consistent with other drawings from this distribution. ∎

Lemma 3.7 *The sum* $\sum_{k=1}^{a_{i,j}} x_{i,k}$ *can be approximated up to a factor of* $1+\epsilon$ *in* $O(\frac{1}{\epsilon}\log a_{i,j})$ *steps.*

Proof: To give a $1+\epsilon$ approximation instead of the above 2-approximation, we imagine rounding each $a_{i,j}$ to the closest value of $\lfloor(1+\epsilon)^s\rfloor$, guaranteeing an answer that is no more than $(1+\epsilon)$ the true value. So we compute sums of the form $\left(\sum_{k=\lfloor(1+\epsilon)^s\rfloor+1}^{\lfloor(1+\epsilon)^{s+1}\rfloor} x_{i,k}\right)$ which is distributed as

$$(\lfloor(1+\epsilon)^{s+1}\rfloor - \lfloor(1+\epsilon)^s\rfloor)^{1/\alpha} S(\alpha, 0)$$

The sum can be computed in $\log_{1+\epsilon} a_{i,j} = \frac{\log a_{i,j}}{\log 1+\epsilon} = O(\frac{1}{\epsilon}\log a_{i,j})$ steps. ∎

The main Theorem 3.1 follows as a consequence of combining Theorem 3.2 with Lemmas 3.5 and 3.7 and by appropriate rescaling of ϵ.

We briefly mention an additional property of this method, which does not follow for the previous method. It is possible to include deletions of values from the past in the following sense: if we are presented with a tuple $(i, -a_{i,j})$, then we interpret this as a request to remove the contribution of $a_{i,j}$ from index i. Provided that there was an earlier tuple $(i, a_{i,j})$, then we can compute the effect this had on $\boldsymbol{z}$ and remove this by subtracting the appropriate quantities. This follows from commutativity and associativity of addition. For similar reasons, it is straightforward to combine the effect of multiple separate readings of signals by combining the data structures for these readings: given two sketch vectors $\boldsymbol{z}_a$ and $\boldsymbol{z}_b$, we can compute the sketch of their combination simply as $\boldsymbol{z}_{a+b} = \boldsymbol{z}_a + \boldsymbol{z}_b$.

4 Hardness of Other Dominances

4.1 Min-Dominance

We recall the definition of the min-dominance, $\sum_i \min_j\{a_{i,j}\}$. We show that, unlike the max-dominance norm, it is not possible to compute a useful approximation to this quantity in the data stream model. This is shown by using a reduction from the size of the intersection of two sets, a problem that is known to be hard to approximate in the communication complexity model.

Theorem 4.1 *Any algorithm to compute a constant factor approximation to* $\mathrm{dom}_{\min}(\boldsymbol{a})$ *requires* $\Omega(n)$ *bits of storage*

Proof: We show a simple transformation to a problem known to be hard in the communication complexity model. Consider two people, A and B, who each hold a subset (X and Y respectively) of the integers 1 to n. Suppose that there were some algorithm which allowed the approximation of the $\mathrm{dom}_{\min}$ function, which is known to both A and B. Then the following interaction takes place: A computes the following data stream $Str(X)$ from the set X:

$$\begin{aligned} Str(X)_i = (i,1) &\iff i \in X \\ Str(X)_i = (i,0) &\iff i \notin X \end{aligned}$$

and passes this stream to the algorithm to process. Following this, A takes the complete memory state of the algorithm and communicates it to B. B then proceeds to run the same algorithm based on this memory state applied to the same transformation of Y, that is, $Str(Y)$. The result is an approximation of $\mathrm{dom}_{\min}$ for the concatenation of the streams $Str(X)$ and $Str(Y)$. But

$$\mathrm{dom}_{\min}(Str(X)||Str(Y)) = |X \cap Y|$$

— in other words, the sum of the minimum values is exactly the intersection size of the two sets. In the communication complexity model, any algorithm that can approximate the size of intersection of two sets to any constant factor with constant probability must communicate $\Omega(n)$ bits of information. This is a consequence of the hardness of the Disjointness problem [KS92, KN97]. Therefore, the size of the memory state of the proposed streaming algorithm must be $\Omega(n)$ bits. ∎

Note that this is as strong a hardness result as we can hope for. The function $\mathrm{dom}_{\min}$ can be computed exactly using $O(n)$ space by storing the stream explicitly. The stream provided has as many convenient properties as we can arrange without making the problem trivial: zero values are represented explicitly, each index occurs exactly twice[3], and all the values seen are either zero or one.

4.2 Quantile and Relative Dominance

Finding the accumulation of any averaging function (Mean, Median or Mode) of a mixture of signals requires as much storage as there are different signals. First, we show the space lower bound on Mean:

Theorem 4.2 *Computing $\sum_i (\sum_j a_{i,j}/n_i)$ on the stream to any constant factor c requires $\Omega(n/c)$ bits of storage.*

Proof: We pick $X \subset \{1 \dots m\}$ such that $|X| = m/2$. We feed $Str(X)$ into the algorithm, and take a copy of the memory contents. We can then pick an arbitrary index k and send a stream of values $\forall i \neq k, 1 \leq j < cm.(i, 0_j)$ to the algorithm. Then

$$\begin{aligned}\sum_i a_i/n_i &= a_k + (m/2 - a_k)/cm \\ &= 1/(2c) \iff a_k = 0 \\ &= 1 + 1/c(1/2 - 1/m) \iff a_k = 1\end{aligned}$$

The ratio of these two quantities is $\frac{1}{2c} : 1 + \frac{1}{c}\left(\frac{1}{2} - \frac{1}{m}\right)$ and if $m \geq 4$ then this is at least $\frac{1}{2c} : \frac{4c+1}{4c} = 1 : \frac{4c+1}{2}$. If we could approximate to better than this ratio, then we could distinguish the two cases. We could retrieve whether any $k \in X$ with constant probability. This would solve the Index problem with one-round communication complexity [KN97], a problem which is known to require $\Omega(m/2)$ communication. Therefore, the memory requirement is $\Omega(m) = \Omega(n/c)$. We are free to set c as large as we like, showing in particular that any constant factor approximation requires $\Omega(n)$ memory use. ∎

We summarize the hardness results for other dominance norms, which are mostly similar to those presented above. Approximating the median or mode dominance ($\sum_i \mathrm{median}_j \{a_{i,j}\}$

[3]Note that if each index occurs at most once then trivially $\mathrm{dom}_{\min}(\boldsymbol{a}) = \mathrm{dom}_{\max}(\boldsymbol{a}) = \sum_i a_{i,1}$

and $\sum_i \text{mode}_j\{a_{i,j}\}$ respectively) requires $\Omega(n)$ space, by a reduction from intersection size. Relative norms, which consist of two distinct stream $\boldsymbol{a}$ and $\boldsymbol{b}$ also prove to be difficult. Approximating the relative sum dominance $\sum a_i/\max\{1, b_i\}$ to any constant c with constant probability requires $\Omega(n/c)$ bits of storage. We omit the proof for brevity: it essentially follows the same pattern as Theorem 4.2. Likewise, approximating the relative count $|\{i|a_i > b_i\}|$ is as hard as approximating set difference size, $|X\backslash Y|$, which also requires $\Omega(n)$ bits. ∎

5 Conclusion

Data streams often consist of multiple signals. We initiated the study of estimating dominance norms over multiple signals. We presented algorithms for estimating the max-dominance of the multiple signals that uses small (poly-logarithmic) space and takes small time per operation. These are the first known algorithm for any dominance norm in the data stream model. In contrast, we showed that related quantities such as the min-dominance cannot be so approximated.

We have already discussed some of the applications of max-dominance, and we expect it to find many other uses, as such, and variations thereof. The question of finding useful indicators for actionable events based on multiple data streams is an important one, and it is of interest to determine other measures which can be computed efficiently to give meaningful indicators. The analysis that we give to demonstrate the behavior of stable distributions with small index parameter α, and our procedure for summing large ranges of such variables very quickly may spur the discovery of further applications of these remarkable distributions.

Acknowledgments

We thank Mayur Datar and Piotr Indyk for some helpful discussions.

References

[AF91] A. Amir and M. Farach. Efficient 2-dimensional approximate matching of non-rectangular figures. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 212–223, 1991.

[AMS96] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 20–29, 1996. Journal version appeared in *Journal of Computer and System Sciences*, 58:137–147, 1999.

[BYJK+02] Z. Bar-Yossef, T.S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisian. Counting distinct elements in a data stream. In *Proceedings of RANDOM 2002*, pages 1–10, 2002.

[BYKS02] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of ACM-SIAM Annual Symposium on Discrete Algorithms*, pages 623–632, 2002.

[CCFC02] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Proceedings of 29th International Colloquium on Automata, Languages and Programming*, 2002.

[CDIM02] G. Cormode, M. Datar, P. Indyk, and S. Muthukrishnan. Comparing data streams using Hamming norms. In *Proceedings of 28th International Conference on Very Large Data Bases*, pages 335–345, 2002. Journal version appeared in *IEEE Transactions on Knowledge and Data Engineering* 15(3):529–541, 2003.

[CIKM02] G. Cormode, P. Indyk, N. Koudas, and S. Muthukrishnan. Fast mining of tabular data via approximate distance computations. In *Proceedings of the International Conference on Data Engineering*, pages 605–616, 2002.

[CMS76] J.M. Chambers, C.L. Mallows, and B.W. Stuck. A method for simulating stable random variables. *Journal of the American Statistical Association*, 71(354):340–344, 1976.

[Cre75] N. Cressie. A note on the behaviour of the stable distributions for small index α. *Zeitschrift fur Wahrscheinlichkeitstheorie und verwandte Gebiete*, 33:61–64, 1975.

[DM02] M. Datar and S. Muthukrishnan. Estimating rarity and similarity over data stream windows. In *Proceedings of 10th Annual European Symposium on Algorithms*, volume 2461 of *Lecture Notes in Computer Science*, pages 323–334, 2002.

[Ele] `http://energycrisis.lbl.gov/`.

[EV01] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of the First ACM SIGCOMM Internet Measurement Workshop (IMW-01)*, pages 75–82, 2001.

[FGL+00] A. Feldmann, A. G. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. In *Proceedings of SIGCOMM*, pages 257–270, 2000.

[FKSV99] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L_1-difference algorithm for massive data streams. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 501–511, 1999.

[FM83] P. Flajolet and G. N. Martin. Probabilistic counting. In *24th Annual Symposium on Foundations of Computer Science*, pages 76–82, 1983. Journal version appeared in *Journal of Computer and System Sciences*, 31:182–209, 1985.

[FS00] J. Fong and M. Strauss. An approximate L^p-difference algorithm for massive data streams. In *Proceedings of the Annual Symposium on Theoretical Aspects of Computer Science*, pages 193–204, 2000.

[GGI+02] A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 389–398, 2002.

[Gib01] P. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *27th International Conference on Very Large Databases*, pages 541–550, 2001.

[GKMS01] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 79–88, 2001. Journal version appeared in *IEEE Transactions on Knowledge and Data Engineering*, 15(3):541–554, 2003.

[GKS01] S. Guha, N. Koudas, and K. Shim. Data streams and histograms. In *Proceedings of Symposium on Theory of Computing*, pages 471–475, 2001.

[GT01] P. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. In *Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures*, pages 281–290, 2001.

[Ind00] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proceedings of the 40th Symposium on Foundations of Computer Science*, pages 189–197, 2000.

[JL84] W.B. Johnson and J. Lindenstrauss. Extensions of Lipshitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[KS92] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

[LSC] Large-scale communication networks: Topology, routing, traffic, and control. `http://ipam.ucla.edu/programs/cntop/cntop_schedule.html`.

[Mut03] S. Muthukrishnan. Data streams: Algorithms and applications. In *ACM-SIAM Symposium on Discrete Algorithms*, `http://athos.rutgers.edu/~muthu/stream-1-1.ps`, 2003.

[Nis92] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12:449–461, 1992.

[Sec] Litigation activities index. `http://securities.stanford.edu/litigation_activity.html`.

[TIGK02] N. Thaper, P. Indyk, S. Guha, and N. Koudas. Dynamic multidimensional histograms. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 359–366, 2002.

[UZ99] V. V. Uchaikin and V. M. Zolotarev. *Chance and Stability: Stable Distributions and their applications.* VSP, 1999.

[YSJ+00] B.-K. Yi, N. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *16th International Conference on Data Engineering (ICDE' 00)*, pages 13–22, 2000.