References for Data Stream Algorithms

Graham Cormode

July 9-11 2007

Abstract

Many scenarios, such as network analysis, utility monitoring, and financial applications, generate massive streams of data. These streams consist of millions or billions of simple updates every hour, and must be processed to extract the information described in tiny pieces. These notes provide an introduction to (and set of references for) *data stream algorithms*, and some of the techniques that have been developed over recent years to help mine the data while avoiding drowning in these massive flows of information.

1 Introduction

In recent years there has been growing interest in the study and analysis of *data streams*: flows of data that are so large that it is usually impractical to store them completely data. Instead, they must be analyzed as they are produced, and high quality results guaranteed, no matter what outcomes are observed as the stream progresses. These notes survey some of the key ideas and techniques that have been developed to analyze and mine such massive data streams. See [Mut05] for a longer survey from an algorithmic perspective. Several other tutorials have covered different aspects of stream processing [BBD+02, GGR02].

Motivation for studying data streams comes from a variety of areas: scientific data generation, from satellite observation to experiments on subatomic particles can generate terabytes of data in short amounts of time; sensor networks may have many hundreds or even thousands of nodes, each taking readings at a high rate; and communications networks generate huge quantities of *meta-data* about the traffic passing across them. In all cases, this information must be processed and analyzed for a variety of reasons: to monitor a system, analyze an experiment, or to ensure that a service is running correctly. However, given the massive size of the input, it is typically not feasible to store it all for convenient access. Instead, we must operate with resources much smaller than the size of the input ("sublinear"), and still guarantee a good quality answer for particular computations over the data.

Data Stream algorithms have been popular since the mid-nineties due to a number of papers introducing and motivating the problems [HRR98, AMS96, FKSV99]. But the earliest algorithms in this model can be traced back about thirty years [MP80, BM81, FM83]. From the disparate motivating settings we can abstract a general framework within which to study them: the streaming model. In fact, there are several variations of this model, depending on what form the input may take and how an algorithm must respond.

1.1 Preliminaries

Models: Arrivals only, or Arrivals and Departures. The basic model of data streams is an arrivals-only one. Here, the stream consists of a quantity of tuples, or items, which describe the input. Typically each tuple is a simple, small object, which might indicate, for example, the

identity of a particular object of interest, and a weight or value associated with this arrival. In a network, the observation of a packet could be interpreted as a tuple indicating the intended destination of the packet, and the size of the packet payload in bytes. For another application, the same packet could be interpreted as a tuple whose identity is the concatenation of the source and destination of the packet, with a weight of 1, indicating that it is a single packet. Typically, we can interpret these streams as defining massive implicit vectors, indexed by item names, and whose entries are (usually) the sum of the associated counts (although many other interpretations may be possible). A richer model allows departures: in additional to positive updates to entries in this implicit vector, they may be negative. This captures more general situations in which earlier updates might be revoked, or observations for which negative values are feasible. In either case, the assumption is that each tuple in the input stream must be processed as it is seen, and cannot be revisited later unless it is stored explicitly by the stream algorithm within its limited internal memory.

Randomization and Approximation. Within these models, many natural and fundamental questions can be shown to require space linear in the input to answer exactly. For example, to test whether two separate streams are the same (i,e. they encode the same number of occurrences of each item) requires us to store space linear in the number of distinct items, which could be immense. To be able to make progress, we typically allow *approximation*: returning an answer that is correct within some small fraction, ϵ of error; and *randomization*: allowing our algorithms to make random choices and to fail with some small probability δ. Algorithms which use both randomization and approximation we refer to as (ϵ, δ) approximations.

Update time, query time and space usage. To evaluate algorithms that operate on streams, we typically look at their behavior with respect to three additional features:

- *Update time:* the time to process each stream update.
- *Query time:* the time to use the information stored to answer the question of interest.
- *Space Usage:* the amount of memory used by the algorithm to keep information.

Typically, these three are measured in terms of parameters of the stream: the number of tuples, n and the number of different items m; and the parameters ϵ and δ. To be an effective streaming algorithm these measures, particularly the space used, should be sublinear in m and n, and ideally poly-logarithmic (i.e. $O((\log m \log n)^c)$ for some constant c.

Tail Inequalities. Many results are proved by defining estimators which are correct in expectation, and then analyzing the probability of being far from the correct answer using tail inequalities, such as Markov, Chebyshev or Chernoff Bounds. These are well covered in standard textbooks [MR95].

1.2 Sampling.

A key problem in data streams is how to draw a sample drawn uniformly from the stream, when the length of the stream is not known in advance (this is also a popular interview question for technical jobs). The research community has developed a rich literature on applications of random sampling algorithms in databases and data streams. The canonical algorithm is often referred to as Reservoir Sampling [Vit85]. One of the most common and well studied applications of sampling in large data warehouse environments is to provide fast approximate answers to complex aggregation queries based on statistical summaries which are created and maintained using various sampling techniques [Olk97, GM98, GMP97]. Random sampling is a standard technique for constructing approximate summary statistics, such as histograms, for

query optimization and query planning purposes [GMP97, CMN98]. Random sampling is widely used for distinct-values estimators [Gib01, GT01, CCMN00] which play an important part in network monitoring and online aggregation systems. Various algorithms for sampling from streams of weighted data items have been proposed in [DLT03, ADLT05, ES06]. Sampling from the support set of distinct items in the stream can be accomplished via min-wise hashing [Bro98, BCFM98], which leads to a variation on reservoir sampling via min-wise sampling [NGSA04].

1.3 Entropy Estimation.

The problem of estimating the entropy of a distribution defined by a stream has attracted a lot of study in the last few years. The problem is to approximate the (zero-th order) entropy of a stream of m values drawn from an alphabet of size n. In the networking world, the problem of approximating the entropy of a stream was considered in Lall et al. [LSO$^+$06]. They focused on estimating F_H, under assumptions about the distribution defined by the stream that ensured that computing H based on their estimate of F_H would give accurate results. Guha, McGregor and Venkatasubramanian [GMV06] gave constant factor as well as (ϵ, δ)-approximations for H, using space that depends on the value of $1/H$. Chakrabarti, Do Ba and Muthukrishnan [CDM06] gave a one pass algorithm for approximating H with sublinear but polynomial in m space, as well as a two-pass algorithm requiring only poly-logarithmic space. Bhuvanagiri and Ganguly [BG06] described an algorithm that can approximate H in poly-logarithmic space in a single pass. The algorithm is based on the same ideas and techniques as recent algorithms for optimally approximating frequency moments [IW05, BGKS06], and can tolerate streams in which previously observed items are removed. The exact space bound is

$$O\left(\epsilon^{-3}(\log^4 m)(\log \delta^{-1})\frac{\log m + \log n + \log \epsilon^{-1}}{\log \epsilon^{-1} + \log\log m}\right) \ ,$$

which is suboptimal in its dependency on ϵ, and has high cost in terms of $\log m$. Cormode et al.[CCM07] gave a sampling-based algorithm which uses $O(\epsilon^{-2}\log(\delta^{-1})\log m)$ words of space, and showed a space lower bound of $\Omega(\epsilon^{-2}/\log^2(\epsilon^{-1}))$, meaning that the algorithm is near-optimal in terms of its dependency on ϵ.

References

[ADLT05] N. Alon, N. Duffield, C. Lund, and M. Thorup. Estimating sums of arbitrary selections with few probes. In *ACM Principles of Database Systems*, 2005.

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

[BBD$^+$02] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *ACM Principles of Database Systems*, pages 1–16, 2002.

[BCFM98] A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *ACM Symposium on Theory of Computing*, pages 327–336, 1998.

[BG06] Lakshminath Bhuvanagiri and Sumit Ganguly. Estimating entropy over data streams. In *ESA*, 2006.

[BGKS06] Lakshminath Bhuvanagiri, Sumit Ganguly, Deepanjan Kesh, and Chandan Saha. Simpler algorithm for estimating frequency moments of data streams. In *SODA*, pages 708–713, 2006.

[BM81] B. Boyer and J. Moore. A fast majority vote algorithm. Technical Report ICSCA-CMP-32, Institute for Computer Science, University of Texas, February 1981.

[Bro98] A. Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of Sequences (SEQUENCES'97)*, pages 21–29, 1998.

[CCM07] A. Chakrabarti, G. Cormode, and A. McGregor. A near-optimal algorithm for computing the entropy of a stream. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[CCMN00] M. Charikar, S. Chaudhuri, R. Motwani, and V. R. Narasayya. Towards estimation error guarantees for distinct values. In *ACM Principles of Database Systems*, pages 268–279, 2000.

[CDM06] Amit Chakrabarti, Khanh Do Ba, and S. Muthukrishnan. Estimating entropy and entropy norm on data streams. In *STACS*, pages 196–205, 2006.

[CMN98] S. Chaudhuri, R. Motwani, and V. Narasayya. Random sampling for histogram construction: How much is enough? In *ACM SIGMOD International Conference on Management of Data*, pages 436–447, 1998.

[DLT03] N. Duffield, C. Lund, and M. Thorup. Estimating flow distributions from sampled flow statistics. In *Proceedings of ACM SIGCOMM*, 2003.

[ES06] P. S. Efraimidis and P. G. Spirakis. Weighted random sampling with a reservoir. *Information Processing Letters (IPL)*, 97:181–185, 2006.

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

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

[GGR02] M. Garofalakis, J. Gehrke, and R. Rastogi. Querying and mining data streams: You only get one look. In *ACM SIGMOD International Conference on Management of Data*, 2002.

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

[GM98] P. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *ACM SIGMOD International Conference on Management of Data*, pages 331–342, 1998.

[GMP97] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. In *International Conference on Very Large Data Bases*, pages 466–475, 1997.

[GMV06] Sudipto Guha, Andrew McGregor, and Suresh Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In *SODA*, pages 733–742, 2006.

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

[HRR98] M. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. Technical Report SRC 1998-011, DEC Systems Research Centre, 1998.

[IW05] Piotr Indyk and David P. Woodruff. Optimal approximations of the frequency moments of data streams. In *STOC*, pages 202–208, 2005.

[LSO+06] Ashwin Lall, Vyas Sekar, Mitsunori Ogihara, Jun Xu, and Hui Zhang. Data streaming algorithms for estimating entropy of network traffic. In *ACM SIGMETRICS*, 2006.

[MP80] J. I. Munro and M. S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12:315–323, 1980.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[Mut05] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Now Publishers, 2005.

[NGSA04] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggrgation in sensor networks. In *ACM SenSys*, 2004.

[Olk97] F. Olken. *Random Sampling from Databases*. PhD thesis, Berkeley, 1997.

[Vit85] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, March 1985.

2 Frequency Moments

Given a stream of items i, let f_i denote the total number of occurrences of i in the stream. The kth frequency moment F_k is defined as

$$F_k = \sum_i |f_i|^k$$

These were studied in the influential paper of Alon, Matias and Szegedy [AMS96].

2.1 F_∞ Estimation.

F_∞ results in the most frequent item. It is provably hard to approximate with relative-error, but good estimates can be found in practice with error bounds stated in terms of F_1 or F_2, via appropriate "sketch" data structures.

The Count-Min (CM) Sketch is a compact summary data structure capable of representing a high-dimensional vector and answering queries on this vector, in particular point queries and dot product queries, with strong accuracy guarantees. Such queries are at the core of many computations, so the structure can be used in order to answer a variety of other queries, such as frequent items (heavy hitters), quantile finding, join size estimation, and more. It also leads to efficient solutions for estimating F_∞ with additive error of at most ϵF_1. Since the data structure can easily process updates in the form of additions or subtractions to dimensions of the vector (which may correspond to insertions or deletions, or other transactions), it is capable of working over streams of updates, at high rates.

The data structure maintains the linear projection of the vector with a number of other random vectors. These vectors are defined implicitly by simple hash functions. Increasing the range of the hash functions increases the accuracy of the summary, and increasing the number of hash functions decreases the probability of a bad estimate. These tradeoffs are quantified precisely below. Because of this linearity, CM sketches can be scaled, added and subtracted, to produce summaries of the corresponding scaled and combined vectors.

The Count-Min sketch was first proposed in 2003 [CM05a] as an alternative to several other sketch techniques, such as the Count sketch [CCFC02] and the AMS sketch [AMS96]. The goal was to provide a simple sketch data structure with a precise characterisation of the dependence on the input parameters. The sketch has also been viewed as a realisation of a counting Bloom filter or Multistage-Filter [EV02], which requires only limited independence randomness to show strong, provable guarantees. The simplicity of creating and probing the sketch has led to its wide use in disparate areas since its initial description.

2.2 F_2 estimation

In their 1996 paper, Alon, Matias and Szegedy gave an algorithm to give an (ϵ, δ)-approximation of the self-join size [AMS96]. The algorithm computes a data structure, where each entry in the data structure is computed through an identical procedure but with a different 4-wise independent hash function for each entry. Each entry can be used to find an estimate of the self-join size that is correct in expectation, but can be far from the correct value. Carefully combining all estimates gives a result that is an (ϵ, δ)-approximation as required. The resulting data structure is often called a "tug-of-war" sketch or AMS Sketch, since the data structure concisely summarizes, or 'sketches' a much larger amount of information.

To build one element of the sketch, the algorithm takes a 4-wise hash function $h : [1 \ldots U] \rightarrow \{-1, +1\}$ and computes $Z = \sum_{i=1}^{U} h(i) f_i$. Note that this is easy to maintain under the turnstile streaming model: initialize $Z = 0$, and for every update in the stream (i, c) set $Z = Z + c * h(i)$.

The value of Z^2 is an unbiased estimate for F_2, due to the independence of the hash functions; further, the variance can be bounded in terms of F_2^2, leading to an efficient (ϵ, δ) approximation via independent repetitions of the estimator.

This sketch can be seen as a low-independence implementation of the Johnson-Lindenstrauss lemma [JL84]. See also [DG99, IM98] for concise explanations and proofs of this dimensionality reduction lemma.

2.3 F_0 estimation

The problem of estimating F_0, also known as distinct counting has been studied in [FM83, Gib01]. This is a foundational problem in database summarization, e.g. consider a traditional database table which is subject to a sequence of insertions and deletions of rows. It is of importance for query optimization to know the number of distinct values that each attribute of the table assumes. The importance of this problem is highlighted in [CCMN00]: "A principled choice of an execution plan by an optimizer heavily depends on the availability of statistical summaries like histograms and the number of distinct values in a column for the tables referenced in the query." Distinct values are also of importance in statistics and scientific computing (see [Gib01, GM99, HNSS95]). Note that it is provably impossible to approximate this statistic without looking at a large fraction of the database (eg via sampling) [CCMN00]. The alternative paradigm consists of synopsis based approaches which keep a small summary of the stream, and update this every time an item is added or removed. The most widely known synopsis method is that of Flajolet and Martin [FM83, FM85]. This approach assumes fully-independent hash functions, although has been noted to work well in practice using standard hash functions. Improved results require only pairwise independence, and can be seen as generalizations of the FM technique [GT01, BYJK$^+$02].

Bar-Yossef et al. [BYJK$^+$02] provide a variety of algorithms for this problem, based on careful analysis of hashing routines. The simplest to describe and analyze simply hashes all items onto a domain of size m^3, and tracks the $t = O(1/\epsilon^2)$ smallest hash values seen. The value of the tth smallest hash value is then used to estimate F_0. This estimator is good with constant probability; repeating several times in parallel with independent random choices drives down the probability of failure exponentially in the number repetitions.

Motivated by sensor networks, generalizations of these data structures have been used to build simple duplicate-resilient aggregates in [CLKB04, NGSA04]. These begin with computing aggregates (SUM, COUNT) in a duplicate-resilient way, and progress to quantiles and heavy hitters. The *Range Efficient* version of the problem occurs when the input arrives as a series of ranges of values $[a \dots b]$, and must be processed efficiently. Pavan and Tirthapura [PT07] provide a range-efficient algorithm based on careful analysis of hash-function structure. This can be used to solve the Dominance Norm problem efficiently [CM03].

2.4 Extensions

Higher Frequency Moments. AMS give a sampling based algorithm for arbitrary F_k with space $O(m^{1-1/k})$ for a stream over a universe of size m [AMS96]. Coppersmith and Kumar give an improved algorithm which is more similar to the F_2 tug of war sketch [CK04]. Optimal bounds are provided by Indyk and Woodruff [IW05] and tightened by Bhuvangiri et al. [BGKS06] based on randomly sampling items at different sampling rates.

Combined Frequency Moments. Cormode and Muthukrishnan proposed the problem of computing multiple frequency moments in the context of dynamic graph streams [CM05b].

These compute frequency moments on top of F_0 on a secondary attribute. Other combinations of frequency moments have not been well studied.

References

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

[BGKS06] L. Bhuvanagiri, S. Ganguly, D. Kesh, and C. Saha. Simpler algorithm for estimating frequency moments of data streams. In *ACM-SIAM Symposium on Discrete Algorithms*, 2006.

[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.

[CCFC02] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Procedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, 2002.

[CCMN00] M. Charikar, S. Chaudhuri, R. Motwani, and V. R. Narasayya. Towards estimation error guarantees for distinct values. In *ACM Principles of Database Systems*, pages 268–279, 2000.

[CK04] Don Coppersmith and Ravi Kumar. An improved data stream algorithm for frequency moments. In *ACM-SIAM Symposium on Discrete Algorithms*, 2004.

[CLKB04] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *IEEE International Conference on Data Engineering*, 2004.

[CM03] G. Cormode and S. Muthukrishnan. Estimating dominance norms of multiple data streams. In *European Symposium on Algorithms (ESA)*, volume 2838 of *LNCS*, 2003.

[CM05a] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[CM05b] G. Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *ACM Principles of Database Systems*, 2005.

[DG99] S. Dasgupta and A. Gupta. An elementary proof of the Johnson-Lindenstrauss lemma. Technical Report TR-99-006, International Computer Science Institute, Berkeley, 1999.

[EV02] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of ACM SIGCOMM*, volume 32, 4 of *Computer Communication Review*, pages 323–338, 2002.

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

[FM85] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for database applications. *Journal of Computer and System Sciences*, 31:182–209, 1985.

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

[GM99] P. Gibbons and Y. Matias. Synopsis structures for massive data sets. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, A, 1999.

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

[HNSS95] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *International Conference on Very Large Data Bases*, pages 311–322, 1995.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing*, pages 604–613, 1998.

[IW05] Piotr Indyk and David P. Woodruff. Optimal approximations of the frequency moments of data streams. In *ACM Symposium on Theory of Computing*, 2005.

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

[NGSA04] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggrgation in sensor networks. In *ACM SenSys*, 2004.

[PT07] A. Pavan and Srikanta Tirthapura. Range-efficient counting of distinct elements in a massive data stream. *SIAM Journal on Computing*, 37(2):359–379, 2007.

3 Lower Bounds

Since its introduction in 1979 by Yao, communication complexity [Yao79, KN97b] has proven to be a powerful technique for proving lower bounds for a variety of computation models including the cell-probe and data-stream models. The majority of results in this area are on the fixed-partition model of communication complexity, where the goal is for two or more players to evaluate a function over their mutual inputs, i.e. computing $f(x, y)$ when one player holds x and the other has y. Many functions can be shown to require a large amount of communication to evaluate when the input is partitioned between the players in this manner. These can imply lower bounds for different models of computation by arguing that such partitions arise in the course of the necessary computation. If we can encode a particular communication "within" a streaming computation, then the communication complexity of the communication problem implies the same space bound for the streaming problem, since the storage used by the algorithm can be seen as a message that is being passed.

Two fundamental "hard" problems in communication complexity that lead to many streaming lower bounds are:

- INDEX: Alice holds string x of length n, Bob holds index $y \in [n]$, and Bob must output the yth bit of x, given a single message from Alice. Even allowing randomization, this problem requires $\Omega(n)$ bits of communication when Alice is allowed to send only a single message. Clearly, when Bob is allowed to talk to Alice first, $O(\log n)$ bits suffice [KN97a].
- DISJOINTNESS: Alice and Bob both hold strings x and y, and must determine whether there is any bit location where both strings are 1. This requires $\Omega(n)$ communication, even allowing multiple rounds of communication and randomization. The problem remains hard even under weakenings, such as the strings being picked uniformly from certain distributions, or a constant fraction of bits being set to 1 [Raz92, KS92].

A third problem is GAPHAMMING: here, Alice and Bob both have strings with the promise that their strings have Hamming distance either less than $N/2-\sqrt{N}$ or greater than $N/2+\sqrt{N}$. This was initially shown to have $\Omega(N)$ communication complexity via a complex VC-dimension argument [IW03, Woo04], but simpler arguments are now known which reduce GAPHAMMING to INDEX [JKS07].

The linear hardness of GAPHAMMING can then be used to show hardness of frequency moments of $\Omega(\epsilon^{-2})$ [IW03, Woo04]; and $\Omega(\epsilon^{-2}/\log(1/\epsilon))$ for entropy estimation [CCM07]. This shows that known algorithms for these problems are essentially optimal in terms of their dependency on ϵ.

References

[CCM07] A. Chakrabarti, G. Cormode, and A. McGregor. A near-optimal algorithm for computing the entropy of a stream. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[IW03] P. Indyk and D. Woodruff. Tight lower bounds for the distinct elements problem. In *IEEE Conference on Foundations of Computer Science*, 2003.

[JKS07] T. S. Jayram, Ravi Kumar, and D. Sivakumar. The one-way communication complexity of gap hamming distance. `http://www.madalgo.au.dk/img/SumSchoo2007_Lecture_20slides/Bibliography%/p14_Jayram_07_Manusc_ghd.pdf`, 2007.

[KN97a] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[KN97b] Eyal Kushilevitz and Noam 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.

[Raz92] A. A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.

[Woo04] D. Woodruff. Optimal space lower bounds for all frequency moments. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 167–175, 2004.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). *ACM Symposium on Theory of Computing*, pages 209–213, 1979.

4 Extensions and Open Problems

Several open problems are detailed in the "Kanpur List" [McG07], although note that some have been addressed since creation of the list.

Deterministic Streaming Algorithms. There are many known deterministic streaming algorithms, especially for problems such as frequent items and quantile finding. See the implementation and comparison paper [CH08] for some detailed discussions of some of these.

Clustering on Data Streams. Formally, given a set of items, a good clustering places those items that are similar together in clusters, and ensures that the items in different clusters are different. It is natural to try to extend clustering to a stream, but what does it mean when the stream is so large we cannot store for each point which cluster it is allocated to? Typically, we seek a number of clusters, k, which is much smaller than the number of points, n to be clustered. After seeing the stream, the output is just the k clusters, from which the mapping of points to clusters is implicit (e.g. each point is mapped to its closest cluster).

An example problem of clustering a stream is to optimize the k-center objective: attempting to minimize the diameter (the maximum distance between any two points in the same cluster). An algorithm arises by guessing the diameter of the clustering is some value d. The first point is allocated a cluster of its own. For each subsequent point in the stream, if it is far from any existing cluster, a new cluster containing the new point is created, else it is allocated to an existing cluster. If the guess of d was good, then no more than k clusters will be created. Moreover, if d was reasonably close to the true diameter, then the diameter of the stream clustering will be within a factor of 2 of the best possible cluster radius. By trying different guesses of d in parallel, and discarding any that generate more than k clusters, we can build a $(1+\epsilon, 0)$ (i.e. deterministic) clustering algorithm [CCFM97, CMZ07]. Different techniques are needed to guarantee good results for other clustering objective functions, such as k-median, k-means and so

Geometric Streams When the stream consists of points, it is natural to estimate geometric quantities such as diameter of the point set, convex hull, etc. [CM03]. In [Ind03, Ind04], certain matching, spanning tree and clustering problems were studied on spatial streams. The work in [KMS02] proposes exponential histograms in one dimension for maintaining statistics on the distribution of points on a straight line. There has been work on estimating diameter of point sets in two dimensions [FKZ02]. Adaptive sampling is introduced in in [HS03, HS04], which approximates the diameter and convex hull of a point set in one pass up to a factor of $1 \pm \epsilon$ in space $O(\epsilon^{-\frac{1}{2}})$.

Other geometric algorithms are based on the notion of "core-sets": a small subset of the input such that solving the problem on the subset gives a good approximation to the solution on the full input [FS05, HPM04, HPM04, Cha06]. And another class of algorithms use a hierarchical approach: solving the problem exactly on a small subset of data that fits in memory, then merging such solutions to get an approximate solution to the full problem.

Sliding Window Computations. The notion of a sliding window is a natural one when processing a stream of updates: since there are too many tuples to store (especially when processing joins), simply drop the oldest tuples. This simple definition holds much complexity, and has led to numerous papers and theses on processing this definition (see [Gol06] and references therein). Evaluating aggregate queries over sliding windows—even simple queries based on sum and count—can require a lot of state to be maintained, since tuples must be stored until they expire to correctly compute their effect on the aggregate. Consequently, there has been much research on approximate computation of aggregates under sliding windows using much smaller space resources. The earliest work focused on tracking sums and counts: both Exponential

Histograms (EH) [DGIM02] and Deterministic Waves [GT02] answer these queries on a window of size N with relative error ϵ by keeping a careful arrangement of $O(\frac{1}{\epsilon}\log \epsilon N)$ counts and timestamps. They can extend to more complex aggregates by replacing their internal counts with other data structures such as sketches, but this causes the space to blow up by further multiples of $\frac{1}{\epsilon}$ and $\log N$.

For more complex functions, such as quantiles and frequent items, Arasu and Manku proposed a generic approach with cost only a $\log \frac{1}{\epsilon} \log N$ factor larger than the unwindowed approximate algorithms [AM04]. Lee and Ting [LT06] reduce the space for frequent items for a fixed size window to $O(\frac{1}{\epsilon})$, the same as the unwindowed case. There has also been recent interest in handling cases where tuples with timestamps do not arrive in timestamp order: results have been shown for sums and counts [BT07], sampling [CTX07] and quantiles and heavy hitters [CKT08a]. This flexibility comes at a cost: the bounds are further logarithmic factors more expensive than their ordered counterparts. Likewise, methods for sampling from a sliding window require space logarithmically (in the number of tuples in the window) larger than the desired sample size [BDM02].

Time-Decay. Among time decay functions, exponential decay is most popular, since a regular counter can be replaced with an exponentially decayed counter without increasing the (asymptotic) space cost. More recently, there has been interest in extending to aggregates beyond sums and counts, including sampling under exponential decay [Agg06], and quantiles and heavy hitters [CKT08b], which obtain the same space bounds as the undecayed case. For decay with other functions, such as a polynomial, the space cost is typically (much) higher. Cohen and Strauss introduced a variety of techniques for tracking sums and counts under backward decay [CS03], with cost $O(\frac{1}{\epsilon}\log N)$. This was extended to sampling and aggregate computation [CTX07, CKT08a], with similar blow-ups of $\text{poly}(\frac{1}{\epsilon}, \log N)$ over the undecayed version.

Other models of streaming (multiple passes, MUD model). A generalized model of streaming allows multiple passes to be made over the data. This is valid for settings where the data may reside on slow storage (e.g. tape) so it can be passed over multiple times, but random-access is not practical. This concept has been explored since the early days of computation, and arguably the first paper on streaming by Munro and Paterson in 1978 [MP80a] looked at pass/space tradeoffs for finding medians. More recently, various other complex problems (graph and matrix streaming) have tolerated multiple passes to improve the space bounds. Feldman et al. [FMS+08] provide a "massive unordered data" model, which abstracts some of the properties of parallel distributed processing settings such as MapReduce, and contrast the power of the model with centralized streaming.

Skewed Streams. Many skewed distributions are well captured by Zipf distributions with appropriate parameters. Natural phenomena, such as sizes of cities, distribution of income, social networks and word frequency can all be modeled with Zipf distributions with parameter z. Even the number of citations of papers demonstrates a highly skewed Zipf distribution [Red98]. More relevant to our study of large data streams, web page accesses for different sites have been observed to obey a skewed Zipf distribution with parameter between 0.65 and 0.8. [BCF+99]. The "depth" to which surfers investigate websites is also captured by a Zipf distribution, with parameter 1.5 [HPPL98]. Files communicated over the Internet display Zipf distribution in a variety of ways: transmission times are Zipf with parameter approximately 1.2; the size of files requested, transmitted, and available for download are all Zipf with parameters respectively measured as 1.16, 1.06 and 1.06 [BCT99]. FTP traffic sizes was estimated to have z in the range 0.9 to 1.1. More strongly, such skewed behavior of requests appears not only over individual addresses but also when grouped into subnets or larger networks [KLPS02], meaning that the skewed distribution is self-similar (multi-fractal).

Sketches have been analyzed in the presence of skewed data. For the top k problem, [CCFC02] specifically studied Zipfian data and showed that for $z > \frac{1}{2}$, $O(\frac{k}{\epsilon^2})$ space suffices. The Count-Min sketch can be used in this setting, an yields a data stream summary that can answer point queries with ϵ accuracy with space only $O(\epsilon^{-\min\{1,1/z\}})$, which is essentially tight for skewed distributions. The same data structure can also estimate F_2 of the stream in $o(1/\epsilon^2)$ space for $z > \frac{1}{2}$ [CM05b].

Graph Streams. In [HRR98], the authors studied special multigraphs and showed $\Omega(n^2)$ lower bounds on space needed for solving many problems. Computing with more general graph properties such as shortest paths, connectivity etc. is provably difficult in the cash register model, so the *semi-streaming* model was proposed [Mut03] where the space is proportional to n, the number of nodes, and sublinear in m, the number of edges. This is of interest in structural analysis of web graphs and in estimating transitivity of binary relations in databases. Some interesting results have been obtained in the semi-streaming model for graph properties such as diameters, spanners etc. [FKM$^+$04, FKM$^+$05]. In [BYKS02], the authors presented a way to estimate the count of triangles in graphs, using space potentially super-linear, but sublinear if the graphs had certain properties.

Matrix Streaming. Streams can define massive matrices explicitly or implicitly. Typically, little can be computed even approximately in a single pass, but in a small constant number of passes, several quantities can be approximated fairly well, in terms of the Frobenius or other norms of the matrix. See the tutorial by Drineas and Kannan for a detailed explanation of approximate sampling and streaming methods for massive matrix analysis [DM06].

Permutation Streaming. Gupta and Zane [GZ03] approximated the number of inversions (disordered pairs of items) in a list. They presented an algorithm that retains the k smallest elements after sampling at an appropriate rate. This is then repeated for every quantile of the form $\frac{1}{\epsilon}(1+\epsilon)^i$ up to n. This gives a total of $\frac{1}{\epsilon}\log \epsilon n$ parallel sampling routines. The $\frac{1}{\epsilon}$ smallest items can be stored exactly. Overall, the space requirement is $O(\frac{1}{\epsilon^3}\log^2 \epsilon n)$ samples.

The problem of approximating the edit distance and longest increasing sequence in a stream has also attracted much study. The most recent results are due to Ergun and Jowhari [EJ08], which reduces the problem to a simpler combinatorial problem (up to a factor of 2), then approximates this using a different generalized quantiles algorithm. A lower bound of $\Omega(N^{1/2})$ is given for the deterministic case, but the randomized setting is less well understood.

Random Order Streams. Random-order data streams were considered in one of the first papers considering a data-stream model [MP80b]. In recent years there has a been a resurgence of interest in the model for a variety of reasons [DLOM02, GMV06, GM06, GM07b, GM07a, CJP08]. Uniform or near uniform orderings can arise in a number of ways, such as when processing a stream of samples that are drawn independently from a non-time varying distribution. For problems such as quantile estimation and finding frequent items it has been shown that there is a considerable difference between processing random-order stream and adversarial streams. In particular, streaming algorithms to find the median using polylog space require exponentially fewer passes if the stream is ordered randomly [GM06] and that this is tight [GM07a, CJP08].

Recent work by Chakrabarti et al. [CCM08] shows that for partitioning t length n binary strings between $\Omega(t^2)$ players, there is an $\Omega(n/t)$ communication bound. For a natural $p+1$ level pointer jumping problem encoded in n bits, any p round protocol between two players requires $\Omega(2^{-p}n^{\frac{1}{(p-1)2^{p+1}-2}})$ bits of communication. This implies the same robust bound for median finding and related selection problems. There are linear lower bounds for the Gap Hamming Distance and Index. These lead to wide variety of lower-bounds for data-stream problems in the random-order model: for approximating frequency moments, the number of distinct values, entropy, information divergences, selection, and graph connectivity.

Probabilistic Streams. Recent work on probabilistic database systems has focused on different aspects of managing uncertain data tuples in relational DBMS architectures, including the complexity and algorithmic problems of query evaluation [DS04], data modeling issues, and managing lineage for probabilistic query results [BSHW06]. Similar questions arise in the processing of streams with probabilistic information. Jayram *et al.* [JKV07, JMMV07] have studied the problem of evaluating simple aggregate functions (focusing, in particular, on `AVERAGE`) over streams of uncertain data. Cormode and Garofalakis studied the expectation and variance of frequency moments on streams of uncertain data [CG07].

Distributed Streams (Sensor nets, P2P). Today, a majority of data is fundamentally distributed in nature. Data for almost any task is collected over a broad area, and streams in at a much greater rate than ever before. In particular, advances in sensor technology and miniaturization have led to the concept of the *sensor network*: a (typically wireless) collection of sensing devices collecting detailed data about their surroundings. A fundamental question is how to monitor such data? The "one shot approach" requires a query to be distributed to relevant sites in the network, and the answer computed and relayed back to the user. Computing simple algebraic functions (e.g., MIN, MAX, SUM, AVG) over a hierarchical architecture can be done using 'in-network' aggregation of fixed-size partial aggregates (e.g., the TAG system [MFHH02]). For more complex functions, techniques avoid linear communication costs through effective approximation using *composable data synopses* for count distinct, quantiles, heavy hitters, join size, and so on [FM83, AMS96, GK04, AGMS99, CM05a, MSDO05].

It is also open to consider extensions to other application areas and more complex communication models, e.g., monitoring P2P services over shared infrastructure (OpenDHT [RBK+05] over PlanetLab), and dealing with constrained communication models (e.g., intermittent-connectivity and delay-tolerant networks (DTNs) [JFR05]).

Continuous, Distributed Computation. The continuous, distributed setting puts a much more stringent demand on us: a continuous query is distributed to the participating sites, and they must collaborate to ensure that the answer to the query is continuously provided to the user that is accurate (e.g., within specified error bounds) compared to the exact current state. Various approaches taken here include:

- **Adaptive Slack Allocation.** A first cut is to take the allowable "slack" in answering the query within allowable bounds, and distribute it between different participants for different query types, e.g., top-k (most frequent) items, [BO03], item values, [OJW03], set expressions, [DGGR04], and duplicate resilient aggregates [CMZ06]. In such settings, communication is necessary to adjust the slacks, plus some global communication is needed when a large rebalancing of slacks takes place.

- **Predictive Models of Site Behavior.** Recent work extends the idea of local-slack allocation by incorporating simple *models* of the data evolution to "predict" site behavior. Combined with intelligent summarization techniques, these approaches only require concise communication exchanges when prediction models are no longer accurate [CGMR05, CG05b, CG05a].

- **Distributed Triggering.** An important common feature of many distributed continuous monitoring problems is evaluating a condition over distributed data and *triggering* when it is met. Recent work that has provided several solutions to this problem based on a variety of techniques, both deterministic and randomized (where the probability of triggering increases with the amount by which a threshold is exceeded) [SSK06, HNG+07, JHRW04, CMY08].

Authenticated Streaming. When the stream is being processed by a third party, what guarantee is there that the correct answer is being returned? Authenticated streaming tries to add additional intermediate information that can be verified if needed. Initial results are due to Garofalakis et al. [GHM07] and Hadjieleftheriou et al. [LYHK07].

References

[Agg06] C. C. Aggarwal. On biased reservoir sampling in the presence of stream evolution. In *International Conference on Very Large Data Bases*, pages 607–618, 2006.

[AGMS99] N. Alon, P. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In *ACM Principles of Database Systems*, pages 10–20, 1999.

[AM04] A. Arasu and G. S. Manku. Approximate counts and quantiles over sliding windows. In *ACM Principles of Database Systems*, 2004.

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

[BCF+99] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *INFOCOM*, pages 126–134, 1999.

[BCT99] A. Bestavros, M. Crovella, and T. Taqqu. *Heavy-Tailed Probability Distributions in the World Wide Web*, pages 3–25. Birkhäuser, 1999.

[BDM02] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 633–634, 2002.

[BO03] B. Babcock and C. Olston. Distributed top-k monitoring. In *ACM SIGMOD International Conference on Management of Data*, 2003.

[BSHW06] O. Benjelloun, A. Das Sarma, C. Hayworth, and J. Widomn. An introduction to ULDBs and the Trio system. *IEEE Data Engineering Bulletin*, 29(1):5–16, March 2006.

[BT07] C. Busch and S. Tirthapura. A deterministic algorithm for summarizing asynchronous streams over a sliding window. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, 2007.

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

[CCFC02] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Procedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, 2002.

[CCFM97] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *ACM Symposium on Theory of Computing*, pages 625–635, 1997.

[CCM08] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Robust lower bounds for communication and stream computation. In *ACM Symposium on Theory of Computing*, 2008.

[CG05a] G. Cormode and M. Garofalakis. Efficient strategies for continuous distributed tracking tasks. *IEEE Data Engineering Bulletin*, 28(1):33–39, March 2005.

[CG05b] G. Cormode and M. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *International Conference on Very Large Data Bases*, 2005.

[CG07] G. Cormode and M. Garofalakis. Sketching probabilistic data streams. In *ACM SIGMOD International Conference on Management of Data*, 2007.

[CGMR05] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *ACM SIGMOD International Conference on Management of Data*, 2005.

[CH08] Graham Cormode and Mario Hadjieleftheriou. Finding frequent items in data streams. In *International Conference on Very Large Data Bases*, 2008.

[Cha06] Timothy M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom.*, 35(1-2):20–35, 2006.

[CJP08] Amit Chakrabarti, T.S. Jayram, and Mihai Pătraşcu. Tight lower bounds for selection in randomly ordered streams. In *ACM-SIAM Symposium on Discrete Algorithms*, 2008.

[CKT08a] G. Cormode, F. Korn, and S. Tirthapura. Time decaying aggregates in out-of-order streams. In *ACM Principles of Database Systems*, 2008.

[CKT08b] Graham Cormode, Flip Korn, and Srikanta Tirthapura. Exponentially decayed aggregates on data streams. In *IEEE International Conference on Data Engineering*, 2008.

[CM03] G. Cormode and S. Muthukrishnan. Radial histograms for spatial streams. Technical Report 2003-11, Center for Discrete Mathematics and Computer Science (DIMACS), 2003.

[CM05a] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[CM05b] G. Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. In *SIAM Conference on Data Mining*, 2005.

[CMY08] Graham Cormode, S. Muthukrishnan, and Ke Yi. Algorithms for distributed, functional monitoring. In *ACM-SIAM Symposium on Discrete Algorithms*, 2008.

[CMZ06] G. Cormode, S. Muthukrishnan, and W. Zhuang. What's different: Distributed, continuous monitoring of duplicate resilient aggregates on data streams. In *IEEE International Conference on Data Engineering*, 2006.

[CMZ07] G. Cormode, S. Muthukrishnan, and W. Zhuang. Conquering the divide: Continuous clustering of distributed data streams. In *IEEE International Conference on Data Engineering*, 2007.

[CS03] E. Cohen and M. Strauss. Maintaining time-decaying stream aggregates. In *ACM Principles of Database Systems*, 2003.

[CTX07] G. Cormode, S. Tirthapura, and B. Xu. Time-decaying sketches for sensor data aggregation. In *ACM Conference on Principles of Distributed Computing (PODC)*, 2007.

[DGGR04] A. Das, S. Ganguly, M. Garofalakis, and R. Rastogi. Distributed set-expression cardinality estimation. In *International Conference on Very Large Data Bases*, 2004.

[DGIM02] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[DLOM02] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Frequency estimation of internet packet streams with limited space. In *European Symposium on Algorithms*, pages 348–360, 2002.

[DM06] Petros Drineas and Michael W. Mahoney. Randomized algorithms for matrices and massive data sets. In *International Conference on Very Large Data Bases*, 2006.

[DS04] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *International Conference on Very Large Data Bases*, 2004.

[EJ08] Funda Ergün and Hossein Jowhari. On distance to monotonicity and longest increasing subsequence of a data stream. In *ACM-SIAM Symposium on Discrete Algorithms*, 2008.

[FKM+04] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*, 2004.

[FKM+05] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. Graph distances in the streaming model: The value of space. In *ACM-SIAM Symposium on Discrete Algorithms*, 2005.

[FKZ02] J. Feigenbaum, S. Kannan, and J. Zhang. Computing diameter in the streaming and sliding-window models. Technical Report YALEU/DCS/TR-1245, Yale University, `http://www.cs.yale.edu/homes/jf/FKZ.pdf`, 2002.

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

[FMS+08] Jon Feldman, S. Muthukrishnan, Anastasios Sidiropoulos, Clifford Stein, and Zoya Svitkina. On distributing symmetric streaming computations. In *ACM-SIAM Symposium on Discrete Algorithms*, 2008.

[FS05] G. Frahling and C. Sohler. Coresets in dynamic geometric data streams. In *ACM Symposium on Theory of Computing*, 2005.

[GHM07] Minos N. Garofalakis, Joseph M. Hellerstein, and Petros Maniatis. Proof sketches: Verifiable in-network aggregation. In *IEEE International Conference on Data Engineering*, 2007.

[GK04] M. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *ACM Principles of Database Systems*, 2004.

[GM06] Sudipto Guha and Andrew McGregor. Approximate quantiles and the order of the stream. In *ACM Symposium on Principles of Database Systems*, pages 273–279, 2006.

[GM07a] Sudipto Guha and Andrew McGregor. Lower bounds for quantile estimation in random-order and multi-pass streaming. In *International Colloquium on Automata, Languages and Programming*, pages 704–715, 2007.

[GM07b] Sudipto Guha and Andrew McGregor. Space-efficient sampling. In *AISTATS*, pages 169–176, 2007.

[GMV06] Sudipto Guha, Andrew McGregor, and Suresh Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 733–742, 2006.

[Gol06] Lukasz Golab. *Sliding Window Query Processing over Data Streams*. PhD thesis, University of Waterloo, 2006.

[GT02] P. Gibbons and S. Tirthapura. Distributed streams algorithms for sliding windows. In *ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2002.

[GZ03] A. Gupta and F. Zane. Counting inversions in lists. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 253–254, 2003.

[HNG+07] L. Huang, X. Nguyen, M. Garofalakis, J. Hellerstein, A. D. Joseph, M. Jordan, and N. Taft. Communication-efficient online detection of network-wide anomalies. In *IEEE INFOCOMM*, 2007.

[HPM04] S. Har-Peled and S. Mazumdar. Coresets for k-means and k-median clustering and their applications. In *ACM Symposium on Theory of Computing*, pages 291–300, 2004.

[HPPL98] B. Huberman, P. Pirolli, J. Pitkow, and R. Lukose. Strong regularities in world wide web surfing. *Science*, pages 95–97, April 1998.

[HRR98] M. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. Technical Report SRC 1998-011, DEC Systems Research Centre, 1998.

[HS03] J. Hershberger and S. Suri. Convex hulls and related problems on data streams. In *Proceedings of Workshop on Manangement and Processing of Data Streams (MPDS)*, 2003.

[HS04] J. Hershberger and S. Suri. Adaptive sampling for geometric problems over data streams. In *ACM Principles of Database Systems*, pages 252–262, 2004.

[Ind03] P. Indyk. Stream-based geometric algorithms. In *Proceedings of Workshop on Manangement and Processing of Data Streams (MPDS)*, 2003.

[Ind04] P. Indyk. Algorithms for dynamic geometric problems over data streams. In *ACM Symposium on Theory of Computing*, 2004.

[JFR05] S. Jain, K. Fall, and P. Rabin. Routing in a delay tolerant network. In *ACM SIGCOMM*, 2005.

[JHRW04] A. Jain, J. Hellerstein, S. Ratnasamy, and D. Wetherall. A wakeup call for internet monitoring systems: The case for distributed triggers. In *Proceedings of the 3rd Workshop on Hot Topics in Networks (Hotnets)*, 2004.

[JKV07] T. S. Jayram, Satyen Kale, and Erik Vee. Efficient aggregation algorithms for probabilistic data. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[JMMV07] T. S. Jayram, Andrew McGregor, S. Muthukrishnan, and Erik Vee. Estimating statistical aggregates on probabilistic data streams. In *ACM Principles of Database Systems*, 2007.

[KLPS02] E. Kohler, J. Li, V. Paxson, and S. Shenker. Observed structure of addresses in IP traffic. In *ACM SIGCOMM Internet Measurement Workshop*, pages 253–266, 2002.

[KMS02] F. Korn, S. Muthukrishnan, and D. Srivastava. Reverse nearest neighbors aggregates over data streams. In *International Conference on Very Large Data Bases*, 2002.

[LT06] L.K. Lee and H.F. Ting. A simpler and more efficient deterministic scheme for finding frequent items over sliding windows. In *ACM Principles of Database Systems*, 2006.

[LYHK07] Feifei Li, Ke Yi, Marios Hadjieleftheriou, and George Kollios. Proof-infused streams: Enabling authentication of sliding window queries on streams. In *International Conference on Very Large Data Bases*, 2007.

[McG07] Andrew McGregor. Open problems in data streams and related topics, IITK workshop on algorithms for data streams '06. `http://www.cse.iitk.ac.in/users/sganguly/data-stream-probs.pdf`, 2007.

[MFHH02] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. In *Proceedings of Symposium on Operating System Design and Implementation*, 2002.

[MP80a] J. I. Munro and M. S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12:315–323, 1980.

[MP80b] J. Ian Munro and Mike Paterson. Selection and sorting with limited storage. *Theor. Comput. Sci.*, 12:315–323, 1980.

[MSDO05] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In *IEEE International Conference on Data Engineering*, pages 767–778, 2005.

[Mut03] S. Muthukrishnan. Data streams: Algorithms and applications. In *ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[OJW03] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *ACM SIGMOD International Conference on Management of Data*, 2003.

[RBK+05] S. Rhea, G. Brighten, B. Karp, J.Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and Y. Harlan. OpenDHT: A public DHT service and its uses. In *ACM SIGCOMM*, 2005.

[Red98] S. Redner. How popular is your paper? An empirical study of the citation distribution. *The European Physical Journal B*, pages 131–134, 1998.

[SSK06] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distribtuted data streams. In *ACM SIGMOD International Conference on Management of Data*, 2006.