DIMACS Technical Report 2004-28
May 2004

Location Streams: Models and Algorithms

by

M. Hoffmann
Department of Computer Science
University of Leicester
Leicester LE1 7RH, UK

S. Muthukrishnan[1,2]
Dept. of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

Rajeev Raman
Department of Computer Science
University of Leicester
Leicester LE1 7RH, UK.

[1]Permanent Member
[2]Supported by NSF ITR 0220280.

ABSTRACT

We propose two new data stream models: the reset model and the delta model. These are suitable for processing data streams comprising locations of moving objects. We present algorithms for tracking the "extent" of such points in both these models that fits the stream constraint of polylogarithmic space and time. Our work adds to the growing knowledge of data stream algorithms in general by initating the study of new models and algoritms for spatial data such as location streams.

1 Introduction

The area of data stream algorithms is just developing. It is the study of algorithmic problems in applications where the data arrives at extremely high speed. Further, the memory available is quite small, so the entire data can not be stored. Thus algorithms have to solve data analyses problems to the best extent they can with sublinear—often polylogarithmic—space, and fast per-item processing. Under these constraints, computing most significant functions on the data stream is provably impossible. So, algorithms typically tend to produce an approximation and are often randomized. Such algorithms have been developed for a number of data analyses problems including estimation of various norms [8, 25], clustering [20, 11], histograms [18], quantiles [19], heavy-hitters [29] etc.

The quintessential application where this arises is in processing internet traffic logs such as the headers on IP packets and IP sessions. Since packets are forwarded at blistering speeds in the internet, these traffic logs are generated at very high speed; memory working at this speed within the routing elements is quite scarce, and the logs have to be processed in real time to solve number of queries involved in monitoring the internet for security breaches, accounting, billing etc. This application has become fairly well developed, with tutorials and a number of research papers in networking conferences [33, 14, 28], specialized database systems aimed at such applications discussed in workshops, tutorials and a number of papers in database conferences [1, 32, 9], and fundamental algorithmic and complexity-theoretic issues being studied in the theoretical computer science community [8, 6, 31].

Recently, there is growing interest in *other* data streams. In particular, an emerging area is one of *spatial* streams. These are location-specific streams. For example, consider a vast collection of moving objects. Over time, their locations change and one wants to track various functions of their locations. GPS (Global Positioning System) enables tracking locations of devices over large scale. Wireless service providers are looking to use this or other infrastructure to enable location-aware services, which is a burgeoning industry [3]. Other industries are actively engaged in developing applications based on tracking the location of various objects: delivery trucks (eg. [2]), University campus buses [4], cars, stocks of goods, biosensors [5], astronomical studies via satellite and ground-based systems [10], etc. See citelbs for a variety of leads.

Motivated by these emerging scenarios of spatial streams, researchers have just begun studying what geometric problems can be solved with high speed spatial streams. In particular, the work thus far has focused on adapting the models from standard data stream context—time series, cash register and turnstile models as summarized in [31]— into the spatial domain, and solving the basic problems such as Diameter estimation [23], Convex Hull estimation [15, 22, 13], Euclidean spanning tree and matching problems [24], Facility location [27], etc.

In our study here, we significantly expand the scope of algorithms for spatial data streams. In particular, we initiate the study of of algorithms for location streams, by following contributions:

1. We formulate two new models of data streams which seem highly relevant for processing

location streams.

2. We study a fundamental problem, ie., monitoring the extent of moving points, under both these models and design space- and time- efficient algorithms for accurately estimating the "extent" of moving objects under different norms.

We believe that our work will lead to a rich area of solving geometric problems in our models, which seem challenging.

As in the study of standard data streams [9, 31], exact calculation of most functions is impossible in our new models, and algorithms tend to be probabilistic and approximate. Our algorithms rely on various random projections to reduce the problem with location data streams to well-studied problems in standard data stream models such as various norm estimation problems. The notion of such reductions itself is generic in Theoretical Computer Science, and has been used in processing spatial data streams as in [13, 24], but the specifics of the reductions are our contributions.

In the remainder of the paper, we will first the models in Section 2. In Sections 3 and 4, we present our results for the two new models. Conclusions are in Section 5.

2 Data Stream Models

Data stream models have been formulated over a series of papers; they are implicit in the succinct data structures described by Gibbons and Matias [16], explicit in [21], and found in its variations in [17].

2.1 Standard Data Stream Models

Input stream $a_1, a_2, \ldots$ arrives sequentially, item by item, and describes an underlying *signal* A, a one-dimensional function $A : [1 \ldots N] \rightarrow R$. Models differ on how a_i's describe A.

- *Time Series Model.* Each a_i equals $A[i]$ and they appear in increasing order of i.

- *Cash Register Model.* Here a_i's are increments to $A[j]$'s. Think of $a_i = (j, I_i)$, $I_i \geq 0$, to mean $A_i[j] = A_{i-1}[j] + I_i$ where A_i is the state of the signal after seeing the ith item in the stream. Much as in a cash register, multiple a_i's could increment a given $A[j]$ over time. This is perhaps the most popular data stream model.

- *Turnstile Model.* Here a_i's are updates to $A[j]$'s. Think of $a_i = (j, U_i)$, to mean $A_i[j] = A_{i-1}[j] + U_i$ where A_i is the signal after seeing the ith item in the stream, and U_i may be positive or negative. This is the most general of the standard data stream models.

Informally, one has the time series model when the data arrives in the order of increasing time, one has the cash register model when items are just inserted into a multiset and one has the turnstile model when items are both inserted and deleted as well. The signal is the distribution of values in the multiset, and we want to evaluate various functions on the signal. (For more variations, see [31].)

Queries call for computing various functions on the signal A as the stream is revealed. There are different performance measures to evaluate algorithms in these models.

- Processing time per item a_i in the stream. (Processing Time)
- Space used to store the data structure on A_t at time t. (Storage)
- Time needed to compute functions on A. (Computing time).

We want to make sure that all of these parameters (processing time, storage and computing time) are small, ie., polylog in the input size. Sometimes in applications one parameter is more crucial than the others. For example, in the networking application, processing time and storage need to minimized aggressively [33], but the computing time may be less critical since it is not encountered at the same rate as the updates.

2.2 New Location Data Stream Models

Work this far with spatial streams has focused primarily on the cash register model [23, 24, 22], and to a limited extent on the turnstile model [13, 27]. That is, algorithms consider the insertion and possibly deletion of points into the multiset that is the signal.

In what follows, we propose two new models suitable for location streams. This is motivated by applications that track moving objects. There are a number of technologies to enable this. Global positioning systems (GPS) provides good accuracy, specially outdoors where one has clear line-of-sight to the satellites. Cellular companies can use their cell tower infrastructure to localize objects. This can be done either in the pull-mode (i.e., the system pings and localizes the moving object) or the push-mode (i.e., the object monitors its own location and can update periodically or as needed).In addition, there are a number of *sensor* technologies that are equipped with compasses, accelerometers, etc., in-built wireless connectivity and computing power so the objects can locate themselves quite well with respect to landmarks. More innovative and powerful sensors will be in the market in the future. All of these methods result in objects being able to locate themselves (i.e., self-position) in space as they move around. As mentioned earlier, an industry of location-aware services is looking to develop applications premised on location streams from moving objects.

We propose two dynamic stream models. In both the dynamic models we consider, there is a collection of n objects, and the current x- and y- coordinates of the i-th object are denoted by x_i and y_i respectively. We assume that $x_i = y_i = 0$ for $i = 1, \ldots, n$ initially, and consider two models of updates.

Reset model: The updates appear in a stream (i, x, y), implying that $(x_i, y_i) := (x, y)$.

Incremental model: The updates appear in a stream $(i, \Delta x, \Delta y)$, implying that $x_i := x_i + \Delta x$ and $y_i + \Delta y$.

The incremental model is quite realistic given the capabilities of sensors and pocket PCs that are commercially available. Even without GPS and other positioning mechanisms, there are sensors that can monitor their acceleration over time and integrate that appropriately to determine the change in their position since the last update. On a pocket PC that is equipped with cellular connectivity, this is quite each to do. (See [26] for example for an operational system with this capability.) The reset model has two independent motivations beyond the location streams context. First, there are many situations where location updates of objects merely corresponds to an update of their current position, without information on their previous whereabouts. This happens for example when mediation location servers update locations of objects and can not keep track of the history of requests. Another example is when location stream actually corresponds to location-dependent queries such as in traffic sensor information systems [30]. Second, there is an independent motivation for this model as self-tuning approach to selectivity estimation in [7]: here, when a query is optimized based only on summaries and run, as a by-product, one gets the precise answer to the query, which can be used to close the feedback loop and improve the summaries. This process is precisely captured by the reset model (for the point queries in databases). Prior work on all of these motivations—localizing objects, self-tuning for selectivity estimation in databases, etc.—is quite extensive, and have only given an illustrative reference for each.

2.3 Problem

The problem we consider is the following. Given a *centre* $c = (c_x, c_y)$, let $\vec{d}$ be the vector of dimension n whose i-th component $d_i = dist((x_i, y_i), c)$, where $dist$ measures distance in the L_p norm for $p \geq 0$.

Our aim is to get information about the "extent" of the points, i.e., distribution of distances. Specifically, we wish to estimate $||\vec{d}||_p$, also for some $p \geq 0$; this gives valuable information about the distribution of distances. If nothing else is mentioned, we assume the centre is the origin.

We believe that estimating $||\vec{d}||_p$ is a fundamental problem in the context of location streams, much as norm estimations have proved to be central to a host of algorithms in the standard data stream context.

The central challenge in solving this problem is the following. As is standard in the streaming context, we assume n is very large. In other words, we have only space sublinear in n that is accessible to the algorithm. As before, we are interested in minimizing the time to process each update and time to compute $||\vec{d}||_p$ when queried.

In what follows, we will solve the problem of estimating $||\vec{d}||_p$ under both the models of location streams under these constraints.

3 Reset Model

In the reset model, we consider the case where the update stream is *monotone*, i.e. an update always causes the distance of an object to increase. We show that:

Theorem 1 *For a monotone update stream, and for any constants $p \geq 0, \delta, 1 \geq \epsilon > 0$ there is an algorithm that returns an estimate $\tilde{d}$ such that $||\vec{d}||_p/(1+\epsilon) \leq \tilde{d} \leq (1+\epsilon)||\vec{d}||_p$ with probability at least $1-\delta$, where distances are measured relative to a fixed centre c. The algorithm takes $O(\tau \log n)$ time to process an update and uses $O(\sigma \log n)$ space, where τ and σ are the time and space required to estimate the number of distinct elements in a stream of items from $\{1,\ldots,n\}$ with an accuracy of $(1+\epsilon/3)^p$ and a failure probability of $O(\delta/\log n)$.*

Proof. Since the centre c is fixed, we ignore the 2-D nature of the updates, and consider the vector $\vec{d}$ as being the underlying vector that is being updated. Specifically, the update (i,x,y), simply sets $d_i := dist((x,y),c)$, and we speak of an update (i,d).

If $p = 0$ then the problem reduces to calculating the L_0 norm of a stream which can be solved as in [12]. Hence we now assume $p > 0$.

We conceptually create an infinite sequence of *buckets*, $\ldots, B_{-1}, B_0, B_1, \ldots$, where the i-th bucket has *weight* $w_i = (1+\epsilon')^i$ and $(1+\epsilon') = (1+\epsilon/3)^p$. At any given time, only a contiguous sequence of buckets $B_{r-s},\ldots,B_r$ is *active*. Associated with each active bucket B_i is the assumed data structure that returns an estimate $\tilde{n_i}$ of the number n_i of distinct elements that are placed into it; more specifically, we require that:

$$\Pr[n_i/(1+\epsilon') \leq \tilde{n_i} < n_i(1+\epsilon')] \geq 1-\delta/(2(s+1))$$

. Activating a bucket merely means allocating memory for such a data structure and appropriately initializing it, deactivating a bucket means freeing the memory used for it.

For any d, we let $\rho(d)$ be the integer k such that

$$\sum_{j=-\infty}^{k-1} w_j \leq d^p < \sum_{j=-\infty}^{k} w_j$$

. It will turn out that if d_{max} is the maximum distance then $r = \rho(d_{max})$. To process an update (i,d), we determine $r_i = \rho(d)$ and distinguish three cases:

- If $r_i < r-s$ we update no buckets.
- If $r-s \leq r_i \leq r$ we place the integer i into buckets $B_{r-s},\ldots,B_{r_i}$.
- If $r_i > r$, we deactivate buckets $B_{r-s},\ldots,B_{r_i-s-1}$, activate any inactive buckets among $B_{r_i-s},\ldots,B_r$, place an the integer i into all these buckets, and finally set $r := r_i$.

To estimate $||\vec{d}||_p$, we proceed as follows. For $j = r-s,\ldots,r$ we obtain an estimate $\tilde{n_j}$ of the number of distinct values placed in B_j and return $\tilde{d} = \left(\sum_{j=r-s}^{r} \tilde{n_j} \cdot w_j\right)^{1/p}$. We now calculate the error in this estimate.

First note that by definition, $\sum_{j=-\infty}^{r_i-1} w_i \leq d_i^p$, and so $\sum_{j=-\infty}^{r_i} w_j \leq (1+\epsilon')d_i^p$. Since $\tilde{n_i} \leq (1+\epsilon')n_i$ for all $i = r-s,\ldots,r$, it follows that with probability at least $1-\delta/2$:

$$\tilde{d} \;=\; \left(\sum_{j=r-s}^{r} \tilde{n_j} \cdot w_j\right)^{1/p}$$

$$
\begin{aligned}
&\leq \left((1+\epsilon')\textstyle\sum_{j=r-s}^{r} n_j \cdot w_j\right)^{1/p} \\
&\leq \left((1+\epsilon')^2 \textstyle\sum_{j=r-s}^{r} d_i^p\right)^{1/p} \\
&\leq (1+\epsilon/3)^2 ||\vec{d}||_p \\
&\leq (1+\epsilon)||\vec{d}||_p
\end{aligned}
$$

To lower-bound the estimate, we now set $s = \lceil 1 + \log_{1+\epsilon'}(n/\epsilon') \rceil$, and note that:

$$
\begin{aligned}
\sum_{j=r-s}^{r_i} w_j &> d_i{}^p - \sum_{j=-\infty}^{r-s-1} w_j \\
&> d_i{}^p - d^p_{max}/(1+\epsilon')^s \\
&\geq d_i{}^p - \epsilon' d^p_{max}/(n(1+\epsilon'))
\end{aligned}
$$

Summing over i, we get that:

$$
\begin{aligned}
\sum_{j=r-s}^{r} n_j \cdot w_j &\geq \sum_{i=1}^{n} d_i{}^p - \epsilon' d^p_{max}/(1+\epsilon') \\
&\geq \left(\textstyle\sum_{i=1}^{n} d_i{}^p\right)/(1+\epsilon')
\end{aligned}
$$

As with the upper bound, we note that the probability that $\tilde{n_i}$ is not a significant underestimate is at least $(1-\delta/2)$; the theorem follows.

In contrast, the problem under the general case when updates are not monotonic is provably impossible to solve under streaming constraints. We can reduce the problem to that estimating $|A| - |A \cap B|$ for two sets A and B given in standard turnstile model, which in turn, can be shown to have a nearly linear space lowerbound for $1+\epsilon$ approximation by known lower bounds in Communication Complexity. We omit showing this process here because it is quite standard once the problem of estimating $|A| - |A \cap B|$ is given as the key.

4 Delta Model

In this model, the problem is formalized as follows. Let $\vec{d}$ be the vector of L_p distances of the points relative to some centre c (the centre can be specified as part of the query). The objective is to be able to estimate $||\vec{d}||_q$ for some q, relative to any centre $c = (c_x, c_y)$. In what follows, an ϵ, δ-estimator of a quantity x is a random variable $\hat{x}$ such that $\Pr[(1-\epsilon)x \leq \hat{x} \leq (1+\epsilon)x] \geq 1-\delta$.

If $p = q$, the problem is equivalent to that of estimating $||\vec{x} - c_x||_p + ||\vec{y} - c_y||_p$, where $\vec{x}$ and $\vec{y}$ are the vectors of x- and y- coordinates respectively. Thus, the problem factors trivially into two 1D L_p norm estimations.

The truly interesting case is when $p = 2$ and $q = 1$; this corresponds to computing the sum of the (Euclidean) distance of the points from a given centre. The results assume that

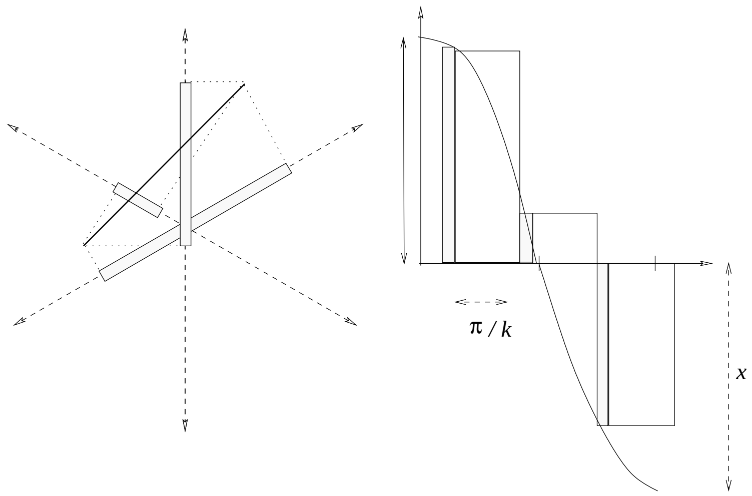

Figure 1: Approximation of a line by its projections.

the choice of centre is independent of the random choices made by the algorithm (more precisely, that the operation sequence is generated by an oblivious adversary).

We begin by considering a simple 1-D version of the problem. Consider a vector $\vec{x}$ of size n that contains x-coordinates, and suppose that we process a sequence of updates $(i, \Delta x)$, which sets $x_i := x_i + \Delta x$. Then, we have that:

Proposition 2 *For any fixed $\epsilon > 0$ and $\delta > 0$, there is a data structure that processes each update in poly-log time, uses poly-log space, and in poly-log time, given any c, returns an ϵ, δ-estimator of $\sum_{i=1}^{n} |x_i - c|$.*

Proof. We essentially observe that the standard algorithm [25] for estimating $||\vec{x}||_1$ also computes the above quantity. The algorithm maintains a *sketch* of $\vec{x}$, which is simply a sequence of values $\vec{x} \cdot \vec{r_i}$, for $i = 1, \ldots, k$, and where r_i is a pseudo-random vector drawn from a *1-stable* distribution. It can be shown that the (normalised) median of $\vec{x} \cdot \vec{r_i}$, for $i = 1, \ldots, k$, for sufficiently large k is an ϵ, δ-estimator of $||\vec{x}||_1$. In order to estimate $||\vec{x} - \vec{c}||_1$, where $\vec{c} = (c, c, \ldots, c)$, we simply compute a sketch of $\vec{x} - \vec{c}$, by computing $\vec{x} \cdot \vec{r_i} - \vec{c} \cdot \vec{r_i}$, for $i = 1, \ldots, k$. This can be done in $O(1)$ time per sketch element provided the sum of the values in $\vec{r_i}$ is known. Since $\vec{r_i}$ is not stored explicitly, this is not trivial, but can be done by using so-called *range-summable* random variables [18, Lemma 2].

Lemma 3 *Let ℓ be a line segment of length x. Consider a collection $\mathcal{C}$ of $k \geq 1$ lines passing through the origin, at angles $(\pi i)/k$ for $i = 0, \ldots, k-1$. Then, if s is the sum of the lengths of the projections of l on all lines in $\mathcal{C}$, then $x(1 - \Theta(1/k)) \leq s\pi/(2k) \leq x(1 + \Theta(1/k))$.*

Proof. Viewing the question as a continuous one, we can approximate the sum by $x \int_0^\pi |\cos(\theta)| d\theta = 2x$. As cos has bounded slope, we have that $|s\pi/k - 2x| = O(x/k)$ (see Figure 1). The lemma follows.

We conclude with the main theorem:

Theorem 4 *For any fixed $\epsilon > 0$ and $\delta > 0$, there is a data structure that processes each update in poly-log time, uses poly-log space, and in poly-log time, given any $c = (c_x, c_y)$, returns an ϵ, δ-estimator of $||\vec{d}||_1$, where distances are the L_2 distances from c.*

Proof. For k a sufficiently large constant, we consider a collection of k lines through the origin as above. Let ℓ be a line in this collection, and project all the points (x_i, y_i) onto ℓ. Let t_i be the distance of the projected point from the origin. We use Proposition 2 to maintain a sketch of the vector $(t_1, \ldots, t_n)$; this allows us also to project a given centre c on to ℓ and use the sketch to estimate the total distance $\tau_\ell(c)$ from the projection of the centre. But $\tau_\ell(c)$ is nothing but the sum of the projections of the distance from c to the points; by Lemma 3, we can sum all estimates of $\tau_\ell(c)$ over ℓ, multiply this by $\pi/2k$, and obtain the desired ϵ, δ estimator.

5 Conclusions and Open Problems

We have proposed two new models for processing high speed location streams, and presented algorithms for the problem of tracking the "extent", i.e., l_p norm of the distances of the points from a center, under both these models.

The problem of processing location streams does not have the immediacy of applications that processing IP network streams has, at this point. But the earliest streaming algorithms were developed around 1996 [8] when the immediacy of IP networking application was not clear. Our models are challenging and we hope they will prove to be a rich source of problems as new applications with location streams become prevalent. Many natural open problems are obvious in the delta and reset models, including the fundamental geometric problems such as estimating convex hulls, etc.

References

[1] DIMACS Workshop on Managing and Processing Data Streams, FCRC 2003. http://www.research.att.com/conf/mpds2003/

[2] http://www.interfleet.com/

[3] http://www.lbszone.com/ http://gislounge.com/ll/lbs.shtml http://www.lbsportal.com/

[4] http://www.whereismybus.com/

[5] http://dimacs.rutgers.edu/Workshops/WGDeliberate/FinalReport5-20-02.doc.

[6] DIMACS Working Group on Streaming Data Analysis. http://dimacs.rutgers.edu/Workshops/StreamingII/

[7] A. Abounaga and S. Chaudhuri. Self-tuning histograms: Building histograms without looking at the data. *Proc. SIGMOD*, 1999.

[8] N. Alon, Y. Matias and M. Szegedy. The space complexity of approximating the frequency moments. *Proc. ACM STOC*, 20–29, 1996.

[9] B. Babcock, S. Babu, M. Datar, R. Motwani and J. Widom. Models and issues in data stream systems. *ACM PODS*, 2002, 1–16.

[10] J. Bates. Talk at NAS meeting on Statistics and Massive Data. http://www7.nationalacademies.org/bms/Massive_Data_Workshop.html.

[11] M. Charikar, L. O'Callaghan and R. Panigrahy. Better streaming algorithms for clustering problems. *ACM STOC*, 2003.

[12] G. Cormode, M. Datar, P. Indyk and S. Muthukrishnan. Comparing data streams using hamming norms (How to zero in). *Proc. VLDB*, 2002, 335–345.

[13] G. Cormode and S. Muthukrishnan Radial Histograms. DIMACS TR 2003.

[14] C. Estan, S. Savage and G. Varghese. Automatically inferring patterns of resource consumption in network traffic. SIGCOMM 2003.

[15] J. Feigenbaum, S. Kannan and J. Ziang. Computing diameter in the streaming and sliding window models. *Manuscript*, 2002.

[16] P. Gibbons and Y. Matias. Synopsis data structures. *Proc. SODA*, 1999, 909–910.

[17] A. Gilbert, Y. Kotidis, S. Muthukrishnan and M. Strauss. Surfing wavelets on streams: One pass summaries for approximate aggregate queris. *VLDB Journal*, 79–88, 2001.

[18] A. C. Gilbert, S. Guha, P. Indyk, P. Indyk, Y. Kotidis, S. Muthukrishnan and M. J. Strauss. Fast, small-space algorithms for approximate histogram maintenance. *Proceedings 34th ACM STOC*, 389–398, 2002.

[19] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. *Proc. ACM SIGMOD*, 2001.

[20] S. Guha, N. Mishra, R. Motwani and L. O'Callaghan. Clustering data streams. *IEEE FOCS*, 2000, 359–366.

[21] M. Henzinger, P. Raghavan and S. Rajagopalan. Computing on data stream. Technical Note 1998-011. Digital systems research center, Palo Alto, May 1998.

[22] J. Hershberger and S. Suri Convex hulls and related problems on data streams. Proc. MPDS 2003.

[23] P. Indyk. Better algorithms for high dimensional proximity problems via asymmetric embeddings. *Proc. ACM-SIAM SODA*, 2003.

[24] P. Indyk. Stream-based geometric algorithms. Proc. MPDS 2003. Proc. STOC 2004.

[25] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. *IEEE FOCS* 2000, 189–197.

[26] R. Jana, T. Johnson, S. Muthukrishnan and A. VItaletti. Location based services in a wireless WAN using cellular digital packet data (CDPD). MobiDE 2001: 74-80

[27] F. Korn, S. Muthukrishnan and D. Srivastava. Reverse nearest neighbor aggregates over data streams. *Proc. VLDB*, 2002.

[28] B. Krishnamurthy, S. Sen, Y. Zhang and Y. Chen. Sketch-based change detection: methods, evaluation and applications. Proc. Internet Measurement Conference (IMC), 2003.

[29] G. Manku and R. Motwani. Approximate frequency counts over data streams. *Proc. VLDB*, 2002, 346–357.

[30] S. Madden and M. Franklin. Fjording the stream: An architecture for queryies over streaming sensor data. *Proc. ICDE*, 2002.

[31] S. Muthukrishnan. Data Streams: Algorithms and Applications. http://www.cs.rutgers.edu/ muthu/stream-1-1.ps

[32] Querying and mining data streams: you only get one look. Tutorial at SIGMOD 2002,VLDB 2002, etc. See http://www.bell-labs.com/user/minos/tutorial.html.

[33] G. Varghese. Detecting packet patterns at high speeds. Tutorial at SIGCOMM 2002.