Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries

Anna C. Gilbert Yannis Kotidis S. Muthukrishnan Martin J. Strauss

AT&T Labs-Research
{agilbert,kotidis,muthu,mstrauss}@research.att.com

Abstract

We present techniques for computing small space representations of massive data streams. These are inspired by traditional wavelet-based approximations that consist of specific linear projections of the underlying data. We present general "sketch" based methods for capturing various linear projections of the data and use them to provide pointwise and rangesum estimation of data streams. These methods use small amounts of space and per-item time while streaming through the data, and provide accurate representation as our experiments with real data streams show.

1 Introduction

Situations abound in which data arrives and is processed in a stream. For example, network service providers collect logs of network usage (telephone calls, IP flows, etc) in great detail from switches and routers into data processing centers, and use them for trend-related analysis. In most cases, not all past history can be accumulated and stored in databases; in cases when data is archived, access to the data is often expensive. Hence it is highly desirable to have approximate, but reasonably accurate, representation of the data stream that can be stored in small amount of space. However, unlike typical selectivity estimation scenarios where such summary data structures are used, it is not realistic to make several passes over the data in the streaming setting. It is crucial that the summary representation be computed on the stream directly, *i.e.*, in one pass.

Consider the following application scenario that arises in telecommunications network monitoring. Switches in telecommunications networks handle a tremendous number of connections every minute. Typically, for each call it handles, a switch dumps a record (known as a Call Detail Record or CDR). These get written when calls complete and, when buffers get full, switches dump them into a central or distributed data processing facility. Eventually, these records flow through the system and get channeled into centers for billing, network operations etc. However, for many mission-critical tasks such as fraud, security and performance monitoring, telecommunications companies need rapid access to the CDRs to perform trend-related analysis: what is the total number of outgoing calls from a telephone number? what is the total traffic at an npa-nxx (the first 6 digits of a telephone number) in the past two hours? Is the outgoing calling pattern of a telephone number unusual? Can a signature be maintained of user profiles? etc. All of these queries need to be answered on the stream since trend analyses are urgent (the sooner a fraud is detected, the sooner it gets stopped). Similar issues arise in monitoring Internet Network elements such as routers, web servers etc. where traffic is potentially far more voluminous.

The need for processing data streams is beginning to be understood, and, consequently, there is effort underway in the data mining [8, 10], database [32] and algorithms [21] communities to address the outstanding problems that arise. Within the database community, it is understood that "*...Today's database systems and data processing algorithms (*e.g.*, data mining) are ill-equipped to handle data streams effectively, and many aspects of data management and processing need to be reconsidered in the presence of data streams.*" [32]. In this paper, we address a fundamental problem that arises in data streaming scenarios, namely, to what extent can the data streams be summarized in small amount of space so accurate estimates can be provided for basic aggregate queries on the underlying signal. While small space data summarization has been studied in the database community recently, data streaming applications present novel issues, chiefly, first, the ability to summarize accurately the signal in one pass, rather than over multiple passes, and, second, the massive scale of the updates to the underlying signal over time.

Our contributions are as follows: We study different data models and formats in data streaming context, and associated aggregate queries that are prevalent. We present a general approach for building small-space, one pass sum-

Proceedings of the 27th VLDB Conference, Roma, Italy, 2001

mary of the signal to enable answering point, range and "aged aggregate" queries of one or more data streams. Our approach is inspired by wavelet transformation methods which are certain linear projections of the signal. If the underlying signal has a small, highly accurate, transform-based representation (as natural signals tend to have), our methods provably provide a high quality approximation. To the best of our knowledge, no such provable results were known before. Our method relies on keeping a "sketch" of the signal so that any linear projection of it can be estimated, and the large projections get estimated provably accurately. Our methods are likely to have further applications for data stream processing, *e.g.*, mining streams using cluster analysis, processing multidimensional streams, etc. We perform an experimental study of our methods for basic and aged aggregate estimation with Call Detail Records drawn from a data stream within AT&T. These results show that our methods capture the underlying signal with a small amount of space very accurately over several hundred million data points.

In Section 2 we discuss related work, while in Section 3 we present different data models and formats for data stream applications and provide the necessary background on wavelet computations. In Section 4 we address some of the theoretical issues in designing algorithms for wavelet transform computation in stream models. In Section 5 we present our approach, together with provable results and address the implementation issues that arise in our sketch-based methods. In Section 6, we present experiments with real data while in Section 7 we present concluding remarks. The proofs of many of our formal claims will be available in the full version of this paper.

2 Related Work

Streaming or one pass algorithms have been studied in different areas. In the area of theoretical algorithms, streaming models have been studied in [21, 3, 12, 13, 22, 20], where methods have been developed for comparing data streams under various L^p distances, or clustering them. Within the database community, one-pass algorithms have been designed for getting median, quantiles and other order statistics [25, 17], correlated aggregate queries [15], mining [14], etc. We focus on summarizing a data stream so that we can accurately estimate individual point and range estimates, a problem not directly considered in the streaming context previously within databases.

Since our work involves small space representation of a data, it relates to selectivity estimation, a well studied topic in databases. Many approaches have been devised in databases for small space approximation of a given function for quick, approximate estimates for point and range queries: sampling techniques [19, 29], histograms [28], wavelet methods [26], etc. A lot of this work is for static data sets where the signal is analyzed off-line to generate summary representation. This does not apply to our data stream scenario. Sampling methods do work for the dynamic case when input is read over time, but they do not directly yield any performance bounds for linear projections that generate wavelet coefficients.

Some amount of work has been done previously on dynamic maintenance of histograms. In [18], the authors show techniques for maintaining an equidepth histogram on the signal. In [1], the authors adopt a learning approach to self-tune any histogram procedure to be more accurate. These techniques maintain a partition of the signal and the central technical difficulty lies in adjusting bucket boundaries, over time. Recently, the problem of maintaining wavelet transforms as data dynamically changes was considered in [27]; in contrast to the problem of maintaining traditional histograms, maintaining wavelet transforms requires tracking significant wavelet coefficients over time, a nontrivial task as the authors argue. When a data item changes in value, many coefficients may get affected and the set of significant coefficients could change rather dramatically (as is revealed from our experiments). Another transform, namely, the Discrete Cosine Transform, was used in [23] where the authors again attempted to maintain the significant transform values over time.

Conceptually, maintaining the set of significant transform values is somewhat similar to the problem of maintaining bestseller items where the goal is to maintain the top-k selling items as sales continue, but could be significantly harder depending upon the transformation. The main difficulty in the hot lists (a special case of what is known as *iceberg queries* [11]) is in detecting which infrequent values become significantly frequent as data items accumulate, using small amount of space. The authors in [23] propose maintaining a *fixed* set of transform coefficients over time. The authors in [27] propose a sophisticated probabilistic counting technique. In either case, no provable results are known on how effective these methods are in tracking the significant coefficients.

There is an effort to study the general principles behind data streaming [32, 33] in database community. To the best of our knowledge, a study of different kinds of data stream models and queries such as aged aggregate queries that we perform in this paper has not previously appeared.

3 Data Streams and Query Processing

In this section, we formally define data streams and different data stream models.

Streaming Data Models. Our input, that we refer to as the *stream*, arrives sequentially, item by item, and describes an underlying *signal*. In the simplest case which we use to develop the notions, the signal a is a one-dimensional function $a : [0 \ldots (N-1)] \rightarrow Z^+$, that is, the domain is assumed to be discrete and ordered[1], and the function maps it to non-negative integers. For example, a signal is the number of employees in different ages (the domain is the set of ages and the range is the number of employees of particular age), or the number of outgoing call minutes

[1] Signals over continuous domains are assumed to be discretized in a sensible fashion, for the purposes of this paper.

from a telephone number (domain is the set of all telephone numbers and the range is the total number of outgoing minutes).

The stream may describe the underlying signal in one of many ways, yielding different data models as a result. There are two distinct models for rendering the signal: *cash register*, or *aggregate models*. In the cash register model, the items that arrive over time are domain values in no particular order, and the function is represented by implicitly aggregating the number of items with a particular domain value. For example, in the telephone calls case, the stream could be:

$$\langle 8008001111, 10\rangle, \langle 8008002222, 15\rangle, \langle 8008003333, 13\rangle, \langle 8008001111, 23\rangle, \langle 8008001111, 3\rangle \ldots$$

The underlying signal, namely $\langle 8008001111, 36\rangle$, $\langle 8008002222, 15\rangle$, $\langle 8008003333, 13\rangle$ has to be constructed by aggregating the total number of minutes outgoing from numbers 8008001111, 8008002222, 8008003333 etc.[2] In the *aggregate model*, the items arrive over time are the range values in no particular order, and the signal is therefore explicitly rendered. For example, in the telephone calls case above the stream could be:

$$\langle 8008001111, 36\rangle, \langle 8008003333, 13\rangle, \langle 8008002222, 15\rangle$$

There are also two distinct formats for the stream: *ordered*, or *unordered*. In the ordered case, the items arrive over time in the increasing (or decreasing) order of the domain values. For example, in the telephone calls case above:

$$\langle 8008001111, 10\rangle, \langle 8008001111, 23\rangle, \langle 8008001111, 3\rangle, \langle 8008002222, 15\rangle, \langle 8008003333, 13\rangle$$

is an example of ordered format in the cash register model. In the *unordered* case, the items that arrive over time are not necessarily directly in the order of the domain values, and may in fact be an arbitrary permutation of the representation.

The two data models and the two data formats jointly give us four possible stream renditions of the underlying signal: ordered/unordered cash register/aggregated renditions. In the cash register model, there is yet another variation depending on whether all the items with a given domain value is *contiguous*. Data streams in the cash register model can be contiguous but not ordered, for example:

$$\langle 8008002222, 15\rangle, \langle 8008001111, 10\rangle, \langle 8008001111, 23\rangle, \langle 8008001111, 3\rangle, \langle 8008003333, 13\rangle$$

Contiguous cash register rendition is equivalent to unordered aggregate rendition under aggregation of the range value for the "running" domain value. Therefore, this rendition is not considered henceforth.

Natural data streams in different application domains may be in different renditions, for example, a time series data is likely to be in ordered aggregate rendition while network volume data is likely to be in the unordered cash register rendition. The unordered cash register model is the most general, posing the most challenges in designing

[2]This model is called the cash register model because the flow of sales through a cash register in a store generates a stream in this model.

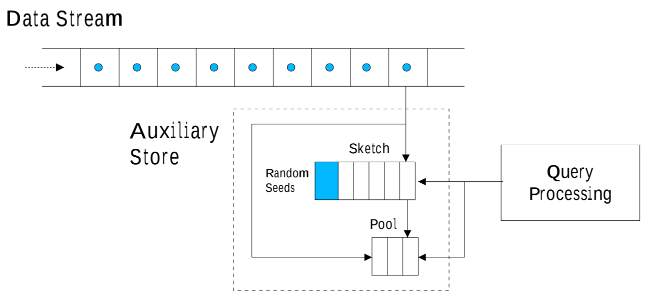

Figure 1: Stream Processing

data processing algorithms. We will present results applicable to all these renditions. For the most part, we focus on the most general one for our upper bounds, and the least general one for the lower bounds, so our results are broadly applicable.

All of the above discussion on data stream models and renditions generalize to multiple signals. For example, a *concatenated stream* is one in which the stream for each signal arrives concatenated one after another. For example, the signal could be the traffic on a telecommunications network from a particular IP address over time for the whole day, and signals for multiple number of days may arrive one after another. Also, the signal may be *multidimensional*, say the (source IP address, destination IP address) aggregation of traffic in networks. As long as all queries are on one of the dimensions, or on all dimensions, or on a subset of dimensions specified *a priori*, the one-dimensional streaming techniques will work. Although our results can be extended to the multidimensional case when this is not the case, additional techniques will be needed, and will not be a subject of this paper.

Stream Processing Model. Now we focus how data streams may be processed. We will present the basic one-pass version of data stream processing. Each data item, as it arrives, is read and processed. No backtracking is allowed on the data stream, and explicit access to arbitrary past items is not permitted. We are allowed a certain amount of additional memory. This may be used, for example, to store a recent window of items we have read or some arbitrary subset of items in the past, or other summary information about the past data stream. The size of this auxiliary store crucially determines what computations can be performed on the data stream. For applications that we consider here, the auxiliary store is significantly smaller than even the signal domain size. Hence, the signal can only be partially represented in the auxiliary store as data items continue to arrive, see Figure 1.

Two performance parameters of our interest are the time needed to process each item on the stream and the auxiliary storage used; our goal would be keep both as small as possible.

Aggregate Queries on Streams. In data stream scenarios, queries are motivated by trend-related analyses. In what follows, we give examples of different types of aggregate queries that tend to be asked, all in the context of telecommunications data. The domain is the telephone number (npa-nxx-line)[3] and the range is the total number of minutes per day of outgoing calls. There is a natural numerical ordering of the domain. Consider the concatenated streams case wherein the signal for each day is concatenated to the previous one but each signal is in unordered cash register model. *In how many minutes of outgoing calls was particular telephone number involved?* This is a typical "point" query on the signal. *How many total minutes of call were handled by a telephone exchange which is given by particular npa-nxx combination?* This is a typical "range" query. An interesting aggregate query in the concatenated data stream scenario is the *aged aggregate query*—from a concatenated data stream $\ldots, a^{(-2)}, a^{(-1)}, a^{(0)}$, where $a^{(0)}$ is a stream of the most recent data (say, today's data), $a^{(-1)}$ is the data from one period before (yesterday's data), etc., define a λ-*aging* data stream to be:

$$\lambda a^{(0)} + \lambda(1-\lambda)a^{(-1)} + \lambda(1-\lambda)^2 a^{(-2)} + \cdots.$$

Thus, recent data contributes to the λ-aging data stream with exponentially more weight than old data. Aggregate queries are posed on the current aged version. Note that aging is not done at the time the data is read, but rather over time, so one can not simply replace the read data item by the final aged data-item.

General Issues. We present techniques to accurately approximate the underlying signal from a stream in the model in Figure 1 that will apply to all aggregate queries above. In evaluating solutions, two parameters of interest are the time it takes to answer a query as well as the accuracy of the answers. We will also assume that we know in advance the size of the domain of i's and a maximum bound for the $a(i)$'s. Our techniques can be extended easily to the case when neither is known in advance.

Background on Wavelet Tranforms. Wavelet transforms [6] (like Discrete Cosine and Fourier transforms) are special mathematical transforms, that attempt to capture the trend in numerical functions. Often, very few of the wavelet coefficients of empirical data sets are significant and a majority are small or insignificant. In practice, a small number of significant coefficients is needed to capture the trends in numerical functions. While the theory of wavelets is extensive, we will only use the rudimentary wavelet transforms in this paper.

We will develop the wavelet background as is typically done using an example computation; see [26] for similar background. Consider the signal of length[4] $N = 8$ given by array $A = [1, 3, 5, 11, 12, 13, 0, 1]$; its Haar wavelet transform computation is shown in Table 1. The transform is computed by convolving the signal with the low pass filter

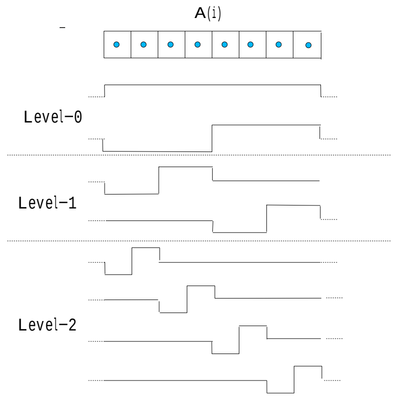

Figure 2: Wavelet-vectors (N=8)

$\{1/\sqrt{2}, 1/\sqrt{2}\}$ and the high pass filter $\{-1/\sqrt{2}, 1/\sqrt{2}\}$, followed by down-sampling by two. In the discrete case if there are N values in the array, this process yields $N/2$ "averages" and $N/2$ "differences" (these are averages and differences respectively, but scaled by a suitable scaling factor). We store the differences as the wavelet coefficients at this level. We then repeat this procedure on the "averages", computing "averages" and "differences" again, until we are left with only one "average" and $N-1$ "differences" over $\log N$ scales or resolutions. The total collection of all the "differences" over the $\log N$ scales together with the final "average" gives the Haar wavelet transform of the input signal. The entire computation can be quite naturally represented by a binary tree over the signal array, each node in the tree representing the "average" of the nodes under it and the "difference" between the left and right child of that node.

The description above of Haar wavelet transforms is illustrative, but not conducive to streaming computations directly, especially when the signal is rendered in unordered cash register model. We will unravel the computation and visualize Haar wavelet transforms in terms of vector computations. Let us number the levels of the binary tree as shown in Table 1 with the bottommost level being 0, and the topmost being $\log N = 3$ in this case. For $j = 1, \ldots, \log N$ and $k = 0, \ldots, 2^j - 1$, define the vector $\phi_{j,k}(l)$ =1 for $k(N/2^j) \leq l \leq k(N/2^j) + N/2^j - 1$, and 0 otherwise. We further define $\psi_{j,k} = -\phi_{j+1,2k} + \phi_{j+1,2k+1}$ for $0 \leq j \leq \log N - 1$ and $k = 0, \ldots, 2^j - 1$. The scaling factor at level j is $s_j = \sqrt{N/2^j}$, for all $j = 0, \ldots, \log N$. Now we can define wavelet vectors to be $s_j \psi_{j,k}$ for each j, k, giving $N-1$ in all. These respectively yield the $N-1$ wavelet coefficients corresponding to the differences given by $d_{j,k} = s_j \langle a, \psi_{j,k} \rangle$ where $\langle x, y \rangle$ is the inner product of vectors x and y. The final "average" is the coefficient that corresponds to the all 1's vector v with scaling factor $s_0 = \sqrt{N}$, that is, $c_{0,0} = s_0 \langle a, v \rangle$; vector $s_0 v$ together with the $N-1$ wavelet vector form the N wavelet basis vectors, see Figure 2.

Formally, we refer to the coefficients ($N-1$ "differences" and one "average") as *wavelet basis coefficients*, and denote them by w_ℓ, so $\{w_\ell : \ell = 0, 1, \ldots, N-1\} =$

[3]Under the North American Numbering Plan, npa is the three digit area code, nxx is the three digit exchange code, and line gives the four digit specific numbering to a telephone in that npa-nxx.

[4]Throughout the exposition, we will assume that N is a power or two. This simplifies notations and discussions without affecting the generality of the results.

1	3	5	11	12	13	0	1
2.8284	11.3137	17.6777	0.7071	**1.4142**	**4.2426**	**0.7071**	**0.7071**
10.0000	13.0000	**6.0000**	**-12.0000**				
16.2635	**2.1213**						

Level $= 3$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
2	$\frac{a_1+a_2}{\sqrt{2}}$	$\frac{a_3+a_4}{\sqrt{2}}$	$\frac{a_5+a_6}{\sqrt{2}}$	$\frac{a_7+a_8}{\sqrt{2}}$	$\boldsymbol{\frac{a_2-a_1}{\sqrt{2}}}$	$\boldsymbol{\frac{a_4-a_3}{\sqrt{2}}}$	$\boldsymbol{\frac{a_6-a_5}{\sqrt{2}}}$	$\boldsymbol{\frac{a_8-a_7}{\sqrt{2}}}$
1	$\frac{a_1+a_2+a_3+a_4}{2}$		$\frac{a_5+a_6+a_7+a_8}{2}$		$\boldsymbol{\frac{(a_3+a_4)-(a_1+a_2)}{2}}$		$\boldsymbol{\frac{(a_7+a_8)-(a_5+a_6)}{2}}$	
0	$\frac{a_1+a_2+a_3+a_4+a_5+a_6+a_7+a_8}{2\sqrt{2}}$				$\boldsymbol{\frac{(a_5+a_6+a_7+a_8)-(a_1+a_2+a_3+a_4)}{2\sqrt{2}}}$			

Table 1: The table shows Haar wavelet decomposition of array $A = [1, 3, 5, 11, 12, 13, 0, 1]$ and the general formula, with entries in the latter shifted horizontally to fit. The wavelet coefficients (*i.e.*, the local differences) are in bold at each level. The final average (16.2635) plus the wavelet coefficients represent the Haar wavelet transformation of the original array.

$\{c_{0,0}\} \cup \{d_{j,k}\}$. Similarly, we refer to the corresponding vectors as *wavelet basis vectors* and denote them by ζ_ℓ, so that $\{\zeta_\ell : \ell = 0, 1, \ldots, N-1\} = \{s_0 v\} \cup \{s_j \psi_{j,k}\}$. That is, $w_\ell = \langle a, \zeta_\ell \rangle$. Hence, informally, wavelet transformation is the inner product of the signal with a specific (rather special) set of N vectors, or equivalently, specific linear projections of the signal. This is the view of wavelet transformations that we adopt henceforth.

Our focus is not on keeping all N coefficients, but rather a much smaller number. In the process, some information about the underlying signal will be lost. Suppose we sort the coefficients, so that $|w_{\ell_1}| \geq |w_{\ell_2}| \geq \cdots$. The *highest* B-term approximation is defined to be $\sum_{k=1}^{B} w_{\ell_k} \zeta_{\ell_k}$, It is easy to derive and it is well known that the highest B-term approximation is in fact the *best* B-term approximation, that is, it minimizes the sum squared error (sse) for a given B.

The *energy* of signal a is defined to be the square of its L_2 norm and is preserved under the wavelet transform *i.e.*, $\sum |a_i|^2 = \sum |w_\ell|^2$.

General Comments. One of the reasons wavelet transformations are popular in engineering, science and financial applications is that most signals that arise in nature have highest (best) B-term approximation with small error for very small values of B, that is, there tends to be a rapidly decaying behavior by which increasing B beyond a small "threshold" does not significantly decrease the sum-squares-error. As an example, Figure 3 plots the sse/energy as a function of B ($1 \leq B \leq 40$) for a day's worth of call detail data. The graph reveals a fast decay in the reduction of the sse as more coefficients are used.

This small-B approximation property motivated the use of wavelets in databases, for similarity search [5] as well as approximate query answering for point and range queries [26, 9, 16]. We were also motivated by this small-B approximation property of wavelets to choose them for data stream processing. However, we are able to exploit this property in two quite distinct ways. First, we use small B to represent the underlying signal to a reasonable approximation. Second, we are able to show how to maintain a small "sketch" of the signal on the stream so that if the original signal had a small B representation which is accurate, then we can generate a possibly different and approximate B-term representation which is nearly as accurate. We are able to do this in a provable manner. This is the basis for our work.

4 Some Foundational Issues

In this section, we will address some of the theoretical issues in designing algorithms for wavelet transform computation in stream models. This section will serve to show the theoretical challenges in designing such algorithms and show the intuition for our approach in the next section.

Let us recall that our goal is to compute the highest (best) B-term approximation to a signal of domain size N. We are working on the data stream that renders this signal. The data stream could be possibly much larger than N depending on the data stream model and the range of the signal.

First, let us consider the ordered aggregate model. Our first theoretical result is a positive one, showing that for the ordered aggregate model, the highest B-term representation can be computed exactly.

Consider the tree representation of the Haar wavelet transformation specified in Section 3. Recall also that in the ordered aggregate model, we get the signal values specified as $(i, a(i))$ in the increasing order of i's. Our algorithm is as follows. Consider reading the data stream and say we are at some position i' in it, that is, we have seen all $a(i)$'s for $i \leq i'$, for some i'. We maintain the following two sets of items:

1. Highest B-wavelet basis coefficients for the signal seen thus far.

2. $\log N$ *straddling* coefficients, one for each level of the Haar wavelet transform tree. At level j, the wavelet basis vector that *straddles* i' is $\psi_{j,k}$ where $k(N/2^j) \leq i' \leq k(N/2^j) + N/2^j - 1$, and there is at most one such vector per level.

When the following data item $(i'+1, a(i'+1))$ is read, we update each of the straddling coefficients at each level if they get affected. Some of the straddling coefficients may no longer remain straddling. When that happens, we compare them against the highest B-coefficients and retain the B highest ones and discard the remaining. At levels in which a straddling coefficient is no longer straddling, a new straddling coefficient is initiated. There will be only one such new straddling coefficient for each level. In this manner, at every position on the data stream, we maintain the highest B-wavelet basis coefficients exactly. This gives,

Theorem 1 *With at most $O(B + \log N)$ storage, we can compute the highest (best) B-term approximation to a signal exactly in the ordered aggregate model. Each new data item needs $O(B + \log N)$ time to be processed.*

In contrast, computing the highest B-term approximation seems to be hard in any other streaming model. Intuitively, keeping track of the highest B numbers in a stream is trivial with the unordered aggregate streaming model (and therefore, the contiguous cash register model), but keeping track of the highest B values of $(c_i - d_i)$ where c_i's and d_i's appear any which way seems to be difficult (and likewise for more complex linear projections like wavelet coefficients). In the unordered cash register model, even keeping track of highest B $a(i)$'s is difficult in general; this is the top-B queries in [11, 4, 2]. We are able to formalize all these intuitions in rigorous mathematical framework and *prove* that computing the highest B-term approximation for a signal in any of these data streaming models is difficult, *i.e.*, would require storing too much data, nearly equal to the size of the signal, and even that of the stream itself! We state our result formally below, but the proof is beyond the scope of this paper.

Theorem 2 *Any streaming algorithm that correctly calculates the highest wavelet basis coefficient (excluding the overall average) of the signal rendered by unordered aggregate stream data (and hence cash register streams) uses $\Omega(N/polylog(N))$ space.*

The strong result above shows that nearly all of the signal must be in the auxiliary store in order to calculate (or even estimate) the highest B-term approximation in data streaming models. This seems to indicate that there is no hope for providing provably good data streaming algorithms for constructing wavelet approximations to the signal. In the next section, we provide an algorithm that gets around this bottleneck by using the small B-term property of wavelet coefficients.

5 Our Data Streaming Algorithms

5.1 Overall Description

We present general techniques for computing wavelet approximations for a signal in data stream models. In what follows, we will describe our overall approach before providing details. All our discussion will be for the most challenging case, namely, the unordered cash register rendition of the signal.

We see the data stream, one item after the other. We maintain a *sketch* of the signal we have seen thus far. The sketch is much smaller than the signal; for a signal over domain of size N, the sketch is of size $\log^{O(1)}(N)$.[5] As data items get read, the sketch gets updated. The sketch has the property that we can generate the linear projections (inner products) of the signal with a small (polynomial) number of vectors quite easily and accurately, provided the dot product of the corresponding unit vectors (the *cosine*) is large. This can be used in several ways. First, since any point query i on the signal can be viewed as merely the inner product of the signal with a vector that has a 1 in its ith component and 0 elsewhere, we can use the sketch to directly estimate the point query; likewise for range queries. Since there are only N point queries and $N(N-1)/2$ range queries which is a small polynomial number, sketches will suffice. Second, since wavelet transforms are linear projections of the signal with a specific set of N vectors, we can generate wavelet coefficient approximations from the sketch which can in turn be used for point or range query estimations on the signal. We will explore both mechanisms, although the latter will prove to be more accurate as our experiments will show.

5.2 Details of Our Approach

Now we will provide the various details, specifically, what is a sketch of a signal, how to compute it on a data stream, and how to use it for estimations.

Sketch and its Computation. Recall that a sketch will be used to estimate the inner product of certain vectors with the signal. We need the following parameters to formally define a sketch and present our claims: ϵ, a *distortion parameter*—we seek inner products correct to within the factor $(1 \pm \epsilon)$ approximation; δ, a *failure probability*—our guarantees will hold with high probability, δ being the failure probability of our claims; η, a *failure threshold*—if the cosine between two vectors is greater than η, we estimate the desired quantity within approximation factor $(1 \pm \epsilon)$ with probability at least $1 - \delta$, but we make no guarantees if the cosine is smaller than η.

An *atomic sketch* of signal a is the dot product $\langle a, r \rangle$, where r is a $\{1, -1\}$ random vector to be defined later. This is the standard random projection approach found, *e.g.*, in [3]. A *sketch* of the signal is $O(\log(N/\delta)/\epsilon^2\eta^2)$ independent atomic sketches, each with a different random vector r.

Since this sketch size is rather small compared to the signal size, we explicitly store the sketch in the auxiliary store. As data stream is read, it is straightforward to update the sketch: when we see an item i in the cash register format, we add r_i^j to the atomic sketch with random vector r^j. In an aggregate format, if we see $a(i)$, we add $a(i)r_i^j$ (which may be rational-valued) to the atomic sketch with random vector r^j. Thus, it is easy to maintain the sketch over a data stream.

An important detail arises, namely, how do we store the random vector r^j's. Notice that the r^j's are of length N

[5]In what follows, $\log^3 N$ will suffice.

each, and explicitly storing them will defeat the purpose of designing small-space sketch on data streams. The seminal idea in [3] is that r^j's can be generated from a seed of size roughly $O(\log N)$ provided the r_i^j's be only 4-wise independent random variables (each r_i^j is *not* generated independently randomly which would make them N-wise independent). Such random vectors are easy to generate as shown in [3].

We adopt this approach, but our requirements on the random vectors r_j's are somewhat more stringent. This is because, later, we will need to estimate the inner product of the signal with the wavelet basis vectors. Each such vector is length N and some of them have N/c 1's for some constant c. Explicitly generating r_i^j for each i with nonzero wavelet basis vector component will thus prove time consuming. What we need is a method to compute these inner products much faster than considering each 1 in the basis vectors. In this paper, we are able to provide such a method.

Our construction of N random variables is novel. It is based on the second order Reed-Muller codes [24]. We describe random variables that take the values 0 or 1; from this construction, simply map $0 \to 1$ and $1 \to -1$. Pick $\log N$ symbols, $a, b, c, \ldots$. The N random variables are indexed by subsets of the symbols. A seed for the random variable is a polynomial over the symbols of degree at most 2, modulo 2, such that each possible term is chosen or not with uniform probability. For example, $1 + b + ac + bc$ is a possible seed. Thus there are $1 + \log N + \binom{\log N}{2}$ possible terms and $2^{1+\log N+\binom{\log N}{2}}$ possible seeds. Using seed s, the value of the random variable indexed by subset A takes the value of s regarded as a polynomial when the symbols in A are set to 1 and the symbols not in A are set to 0. For example, if $A = \{b, c\}$, then, under seed $1 + b + ac + bc$, the A'th random variable takes the value $1 + 1 + 0 \cdot 1 + 1 \cdot 1 = 1 \bmod 2$, and is mapped to -1 as a ± 1-valued random variable.

The construction above has the property that (1) the random variables are 7-wise independent, (2) from seed s for random vector r^j and index i, we can generate r_i^j quickly, and (3) from seed s for r^j, one can find the dot product $\langle r^j, \zeta \rangle$ quickly for any wavelet basis vector ζ. In both (2) and (3) above, quickly means in time polylogarithmic in N. Property (1) will be used in proving our main claim about wavelet approximations using a sketch. All of the above will form part of the full version of this paper.

Using a Sketch to Compute Signal Estimates. A sketch can be used to estimate the signal in two ways described earlier, namely, direct estimation of point queries or estimation of highest B-coefficients. We describe each method below.

For both methods, we need a technical primitive, namely, estimating the inner product of two vectors given their sketches. Given two vectors, a and b, let $\alpha = a/\|a\|_2$ and $\beta = b/\|b\|_2$ be their unit vectors. The inner product $\langle \alpha, \beta \rangle$ can be expressed as $1 - d^2(\alpha, \beta)/2$, where $d(\alpha, \beta)$ is the Euclidean distance between α and β. Given a sketch, A, consisting of atomic sketches, A_j, for stream a, we can estimate the norm squared $\|a\|_2^2$ to $(1 \pm \epsilon)$ [3]: take the median of $O(\log(N/\delta))$ copies of averages of $O(1/\epsilon^2)$ copies of squares of atomic sketches, $\mathrm{median}_{O(\log(N/\delta))} \mathrm{mean}_{O(1/\epsilon^2)} A_j^2$. Similarly, we can estimate $\|b\|_2^2$. By linearity of the sketching technique [3], form $A/\|a\|_2$ and $B/\|b\|_2$, which are sketches for α and β. Next, estimate the squared distance $d^2(\alpha, \beta)$ to $(1 \pm \epsilon\sqrt{\epsilon\eta/B})$ between these two unit vectors by taking the median of $O(\log(N/\delta))$ copies of averages of $O(B/(\eta\epsilon^3))$ copies of $(A_j/\|a\|_2 - B_j/\|b\|_2)^2$, where A_j and B_j are corresponding atomic sketches for a and b. Compute the cosine as $1 - d^2(\alpha, \beta)/2$, and, finally, multiply by $\|a\|_2 \cdot \|b\|_2$ to scale the inner product back from unit vectors to the original. By [3], we estimate $\langle a, b \rangle$ as $\langle a, b \rangle \pm \epsilon\sqrt{\eta\epsilon/B}\|a\|_2\|b\|_2$. Thus, if $\langle a, b \rangle^2 \geq (\eta\epsilon/B)\|a\|_2^2\|b\|_2^2$, our estimate of $\langle a, b \rangle^2$ is good to within the factor $(1 \pm O(\epsilon))$.

5.3 Answering Queries from the sketch.

We consider two variations of query processing: batched and adhoc. In the *batched* model, queries may only be posed at periodic intervals, for example, after the end of the day; hence, queries need not be answered mid-stream. We can perform some time-consuming additional processing during the batching period, since it can be amortized against the entire input stream and the collection of queries. In the *adhoc* model, queries may be posed at any time during the stream processing and rapid response is desirable.

Batched Query Processing. We can use the techniques of section 5.2 to compute an approximation to the highest B-term representation using the sketch. We proceed by estimating each wavelet coefficient as described[6] and selecting up to B of the largest coefficients, but only those whose square is greater than $(\eta\epsilon/B)\|a\|_2^2$ (in practice, we take the B largest coefficient estimates). We use the estimates as coefficients in an at-most-B term approximation to the signal. When a point or range query arrives, using standard wavelet techniques, answer the query in time $O(B \log N)$ [26]. To summarize, we define the energy of a representation R to be $\|a\|_2^2 - \|a - R\|_2^2$, and get

Theorem 3 *There is a streaming algorithm, A, such that, given a signal $a[1..N]$ with energy $\|a\|_2^2$, if there is a B-term representation with energy at least $\eta\|a\|_2^2$, then, with probability at least $1 - \delta$, A finds a representation of at most B terms with energy at least $(1-\epsilon)\eta\|a\|_2^2$; otherwise, A reports "no good representation." In any case, A uses*

$$O\left(\log^2(N)\log(N/\delta)B/(\eta\epsilon^3)\right)$$

space and per-item time while processing the stream. This holds in both the aggregate and cash register formats.

[6]Note that a wavelet basis vector is already normalized, so one does not need to normalize it explicitly.

day	number of records
0	45,110,132
1	81,546,187
2	98,820,613
3	96,768,015
4	97,141,335
5	41,285,628
6	50,361,885
Total	511,033,795

Table 2: Dataset sizes

Adhoc Query Processing. Now we show how to estimate point queries directly using the sketch. We give two methods; which method to use may depend on the data distribution. For a point query i, associate with it the vector e_i consisting of a 1 in position i and zeros elsewhere; an atomic sketch of e_i with random vector r^j is simply r_i^j. We compute the sketch of e_i and the sketch of a to estimate $\langle e_i, a\rangle = a_i$. If $a_i^2 \geq \eta\|a\|_2^2$, then our estimate will be $a_i(1 \pm \epsilon)$ except with probability δ/N. If we make a total of N such queries (not counting repeated queries about the same i) then all of them will be approximately correct except with probability δ. Notice that the procedure above works for not only point queries, but also for any range query or wavelet coefficient since they correspond to computing the inner product distance with a vector. Another way to answer point query i directly from the sketch is to estimate all $\log N$ wavelet basis coefficients w_ℓ that involve a_i, and sum the i'th components, from each of the corresponding vectors $w_\ell \zeta_\ell$ such that $w_\ell^2 \geq \eta\|a\|_2^2$. Use the techniques described above to estimate large coefficients.

Details of our Implementation. A caveat of finding an estimate for the best B-term approximation (for batch query processing) is that the above implementation takes time $\Omega(N \log N)$ which may be prohibitive in applications with large N. In practice, we implement our query processing engine as follows. We maintain the sketch of the signal as well as a *pool* of B coefficients. When a new data item is read, we update the sketch as well as the pool of coefficients. Periodically, we cycle through the set of N wavelet basis vectors and estimate a batch of their coefficients and update the pool to contain the highest B coefficient estimates in all. This way we amortize the cost of computing the estimates against that of reading new data items. The entire implementation needs $O(B + \log^{O(1)} N)$ space for the sketch, the pool as well as all requisite seeds of random vectors.

5.4 Extending Our Approach to Concatenated Streams

Suppose a new stream arrives daily on the same domain, for example, in the domain of phone numbers, the start times and originating phone number of all phone calls made on that day arrives. A data analyst may want to learn approximately the average number of phone calls per day made "recently" by a particular phone number. In practice, the number of calls made in the last day is not useful, since the last day may be unusual; on the other hand, the average number of calls made over the last year may contain other than "recent" behavior.

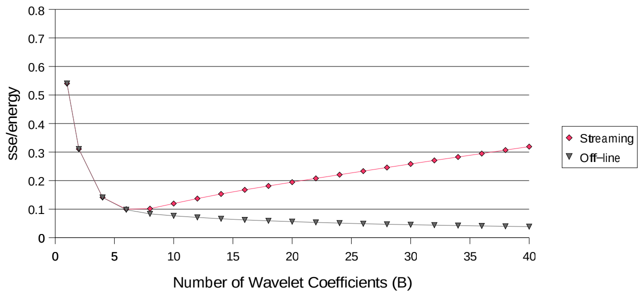

Figure 3: Ratio sse/energy v.s. B for day 0

One approach to this problem is to keep a sliding window of, say, the calls made in the last 30 days; queries are made against this data set. When a new day's data arrives, the oldest day's data is discarded. Thus, this technique requires 30 times more storage than storing a single day's data.

Another approach [7], used in practice at AT&T, is to use λ-aging data. We (inductively) maintain

$$b = \lambda a^{(-1)} + \lambda(1-\lambda)a^{(-2)} + \cdots + \lambda(1-\lambda)^j a^{(-j)} \cdots.$$

When $a^{(0)}$ arrives, replace b with $\lambda a^{(0)} + (1-\lambda)b$. This way, data from j periods ago affect b with exponentially-small relative weight of $(1-\lambda)^j$, so queries against b reflect "recent" activity, but the storage requirement is roughly comparable to storing a single day's data, not 30 days' data.

Note that the foregoing discussion applies to full data sets. We now show that our sketching techniques support queries from λ-aging data sets.

Our sketches are *linear*, that is, from sketches $h(a^{(0)})$ and $h(b)$ of $a^{(0)}$ and b respectivelly, we can form $h(\lambda a^{(0)} + (1-\lambda)b) = \lambda h(a^{(0)}) + (1-\lambda)h(b)$. Note that the weight ultimately assigned to a data point depends on the time of a query and may be different for several queries. Thus, it is not (obviously) possible to modify data as it enters a data structure to simulate a λ-aging data set.

6 Experiments

For the experiments presented in this section we obtained traces from AT&Ts call detail data for a period of one week. The dataset describes a certain type of long-distance calls, aggregated at the npa-nxx level. The stream was in an unordered cash-register format, being an unordered sequence of npa-nxx values, one for each phone-call of that particular type made, where the npa-nxx value corresponds to the originating number. Table 2 shows the number of records for each one of the 7 days of the week as well the total aggregate.

For the first experiment we used the data-feed for day-0 (45M records) and computed off-line the highest-B wavelet coefficients for $1 \leq B \leq 40$. For the streaming set-up, the distortion ϵ and failure threshold η where both set to

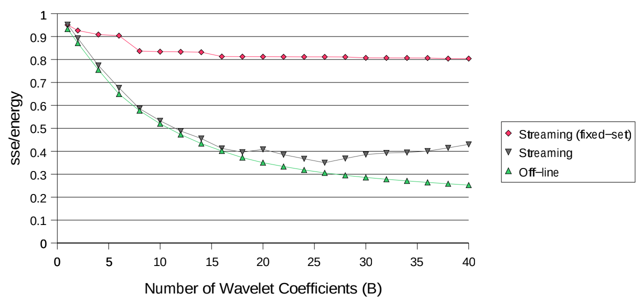

Figure 4: Ratio sse/energy v.s. B at the end of day 6

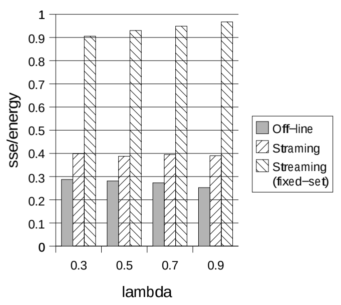

Figure 5: Performance on λ-*aged aggregate queries*

0.3; thus, we were expecting to compute coefficients additively to within $\pm 9\%$ of the energy of the signal. The sketch size for these parameters was 3,952 words long. In the end of the day, we used the sketch to compute a highest-B approximate set of wavelet coefficients for B between 1 and 40. We then reconstructed the signal from both highest-B sets (varying B) and computed the point-wise sum-square error (sse) of each approximation; that is, the cumulative sum-square error of all point-queries on the npa-nxx domain. Figure 3 plots the sse of both representations over the energy of the signal, varying B. The highest-7 wavelet coefficients are accurately picked by the sketch as they contain (cumulatively) roughly 91% of the energy. For $B > 7$ additional wavelet coefficients contain too little information to be reliably identified by the sketch. This is seen in the fact that the ratio for the off-line case flattens-out after that point.

For the next experiment we used data from all 7 days. We compare the highest-B set of coefficients obtained from the sketch with (i) the highest-B selection obtained from an off-line algorithm and (ii) a static highest-B set that is obtained by picking the best-B coefficients after looking at day-0 and dynamically maintaining these coefficients in a streaming fashion, as described in [26]. The latter method is denoted as *Streaming (fixed-set)*. Figure 4 plots the ratio of the sse (for point-queries) over the energy of the signal for all three algorithms varying B. The queries were ran after the end of day-6 (*e.g.* after all 511M records had been processed). This time the sketch managed to pick out most of the good highest-40 wavelet coefficients. Again there is a decrease in the accuracy for $B > 26$ as the remaining coefficients are too small to be computed from the sketch with good accuracy. This streaming model, however, by far outperforms the case when the highest-B selection of coefficients is fixed.

In order to check whether the static selection of coefficients was hindered because of a poor choice of the initial data (day-0) we re-run the experiment starting from day-1 with similar results.

6.1 Processing λ-aging Streams

We futher tested λ-*aged aggregate queries*, varying λ between 0.3 and 0.9. Higher values of λ emphasize the recent history more strongly. Figure 5 shows the ratio of the sse of all point queries over the energy of the λ-aged signal, computed at the end of day-6 after all data had been processed. Again, streaming wavelets are very close to the error obtained by an off-line processing of the stream.

6.2 Updating Wavelet Coefficients in the Background

To find the highest-B coefficients from the sketch, all N coefficients need to be estimated, which may be prohibitive in applications were adhoc queries are expected during the streaming process.[7] In an additional experiment, we used a pool of $B = 40$ coefficients and amortized the cost of estimating the wavelet coefficients by computing $10,000$ coefficients in the background every $10,000,000$ items (*e.g.* we amortize the cost of a coefficient over 1,000 data items). Figure 6 shows the performance of all methods for executing all possible point queries, at the end of each day (starting from day-1). For this set-up, each wavelet coefficient was estimated about 7 times during the 6-day run of the experiment. As more and more wavelet coefficients are updated, the background computation catches up—and sometimes even surpasses—the batch computation of all coefficients. This is a result of different levels of “noise” introduced from the sketch at the time of the wavelet computation.

During these runs, we also computed the sse of answering a point aggregate query directly from the sketch, without using the wavelet coefficients. However, this method repeatedly produced the worst approximation and is not included in the graphs.

7 Conclusions

In this paper, we addressed a fundamental question in the data streaming context, namely, how to summarize the signal represented by the stream in small space so that aggregates queries on the signal can be answered with reasonable accuracy. We present general methods for solving this problem based on storing a sketch of the signal from which many linear projections of the signal can be generated. In particular, we are able to obtain high quality approximations for the wavelet transform of the signal. Using real data from AT&T call detail records, we show our methods to be very effective.

[7]For this dataset, computing a single wavelet coefficient from the sketch takes about 22msecs in a 700Mhz Pentium III PC.

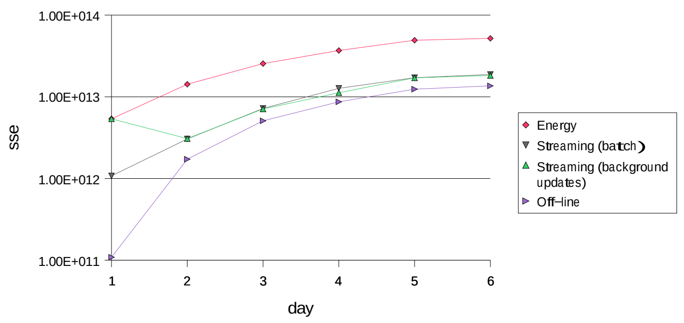

Figure 6: sse at the end of each day, starting from day-1

Our methods are more general than the context in which we have explored them. For example, one of the attractive features of wavelet based methods is that (1) they scale well for multi-dimensions unlike traditional selectivity estimation methods [30], and (2) they have been found to work for data-cube approximations as well [31]. Our methods can be extended quite naturally to those contexts. Also, recently, the notion of correlated or continuous queries has been explored for data streams [15]; we believe that our methods would supplement those results and enhance them to include generalized correlations. Finally, there is some focus on developing data mining algorithms for streams. Such algorithms would need to be able to compare parts of the stream with others repeatedly; hence, they would need small space methods to approximate the distance between "substreams" and "subsignals" efficiently. Our methods may prove useful there.

References

[1] A. Aboulnaga and S. Chaudhuri Self-tuning Histograms: Building Histograms Without Looking at Data. In *Proceedings of the ACM SIGMOD Conference*, June 1999.

[2] N. Alon, P. Gibbons, Y. Matias and M. Szegedy. Tracking join and self-join sizes in limited storage. In *ACM Symposium on Principles of Database Systems (PODS)*, 1999.

[3] N. Alon, Y. Matias and M. Szegedy The Space Complexity of Approximating the Frequency Moments. In *ACM Symp on Theory of Computing (STOC)*, pages 20–29, 1996.

[4] S. Chaudhuri and L. Gravano. Evaluating Top-k Selection Queries. In *Proceedings of VLDB Conference*, 1999.

[5] K. Chan and A. Fu. Efficient Time Series Matching by Wavelets. In *Proceedings of ICDE*, pages 126–133, 1999.

[6] C. K. Chui. An Introduction to Wavelets *Wavelet Analysis and its Applications*, Vol 1, Academic Press, 1992.

[7] C. Cortes and D. Pregibon. Signature-based Methods for Data Streams. *KDD*, to appear.

[8] C. Cortes, K. Fisher, D. Pregibon, A. Rogers and F. Smith. Hancock: A Language for Extracting Signatures from Data Streams. In *KDD*, pages 9–17, August 2000.

[9] K. Chakrabarti, M. Garofalakis, R. Rastogi and K. Shim. Approximate Query Processing Using Wavelets. In *Proceedings of VLDB*, pages 111–122, September 2000.

[10] P. Domingos and G. Hulten. Mining High-Speed Data Streams, P. Domingos, G. Hulten. In *KDD*, August 2000.

[11] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani and J. Ullman. Computing Iceberg Queries Efficiently. In *Proceedings of VLDB Conference*, pages 299–310,1998.

[12] J. Feigenbaum, S. Kannan, M. Strauss and M. Viswanathan. An Approximate L^1-Difference Algorithm for Massive Data Streams. In *FOCS*, pages 501–511, 1999.

[13] J. Feigenbaum, S. Kannan, M. Strauss and M. Viswanathan Testing and Spot-Checking of Data Streams. In *SODA*, pages 165–174, January 2000.

[14] V. Ganti, J. Gehrke and R. Ramakrishnan. Mining Very Large Databases. In *IEEE Computer* 32(8), 1999.

[15] J. Gehrke, F. Korn and D. Srivastava. On Computing Correlated Aggregates Over Continual Data Streams. In *Proceedings of the ACM SIGMOD Conference*, May 2001.

[16] A. Gilbert, Y. Kotidis, S. Muthukrishnan and M. Strauss. Optimal and Approximate Computation of Summary Statistics for Range Aggregates. In *Proceedings of PODS*, pages 227–236, May 2001.

[17] M. Greenwald and S. Khanna. Space-Efficient Online Computation of Quantile Summaries. In *SIGMOD* , 2001.

[18] P. Gibbons, Y. Matias and V.Poosala. Fast Incremental Maintenance of Approximate Histograms. In *Proceedings of VLDB*, Athens, Greece, pages 466–475, August 1997.

[19] P. Gibbons and Y. Matias. New Sampling-Based Summary Statistics for Improving Approximate Query Answers. In *Proceedings of the ACM SIGMOD Conference*, June 1998.

[20] S. Guha, N. Mishra, R. Motwani and L. O'Callaghan. Clustering Data Streams. In *FOCS*, pages 359–366, Nov. 2000.

[21] M. Henzinger, P. Raghavan and S. Rajagopalan. Computing on Data Streams. DEC SRC TR 1998-011, 1998.

[22] P. Indyk. Stable Distributions, Pseudorandom Generators, Embeddings and Data Stream Computation. *FOCS*, 2000.

[23] J.Lee, D. Kim and C. Chung. Multi-dimensional Selectivity Estimation Using Compressed Histogram Informantion In *Proceedings of the ACM SIGMOD Conference*, June 1999.

[24] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes*. North Holland Mathematical Library, Vol. 16, North Holland, New York, 1977.

[25] G. Manku, S. Rajagopalan and B. Lindsay. Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Datasets. In *SIGMOD*, 1999.

[26] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-Based Histograms for Selectivity Estimation. In *SIGMOD* , 1998.

[27] Y. Matias, J. Vitter and M. Wang. Dynamic Maintenance of Wavelet-based Histograms. In *Proc. of VLDB*, Sept. 2000.

[28] V. Poosala. *Histogram-Based Estimation Techniques in Database Systems.* Ph. D. dissertation, University of Wisconsin-Madion, 1997.

[29] Y. Wu, D. Agrawal and A. Abbadi. Using the Golden Rule of Sampling for Query Estimation. In *SIGMOD*, May 2001.

[30] J. Vitter and M. Wang. Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. In *Proceedings of ACM SIGMOD Conference*, June 1999.

[31] J. Vitter, M. Wang and B. Iyer. Data Cube Approximation and Histograms via Wavelets. In *CIKM*, November 1998.

[32] http://www-db.stanford.edu/stream/

[33] http://www.cs.cornell.edu/database/Himalaya/