One-Pass Wavelet Decompositions of Data Streams

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

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 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.

Index Terms—Data streams, wavelets, randomized algorithms, approximate queries.

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 detail history can be accumulated and stored in online databases; often past data is archived and access to it is very expensive. Hence, it is highly desirable to have an approximate, but reasonably accurate, representation of the data stream that can be stored in a 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 generates 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 such as: What is the total number of outgoing calls from a telephone number? What is the total traffic at an npa-nxx (the first six 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? 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 and Web servers 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 [11], database, and algorithms 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.*"[1] 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 a small amount of space so that 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.

In Section 2, we discuss related work, while in Section 3 we present different data models and formats for data stream applications. In Section 4, we provide the necessary background on wavelet computations. In Section 5, we address some of the theoretical issues in designing algorithms for wavelet computations in data streams. In Section 6, we present our approach, together with provable results and address the implementation issues that arise in our sketch-based methods. In Section 7, we present experiments with real data, while in Section 8 we present concluding remarks.

1. http://www-db.stanford.edu/stream.

• *The authors are with AT&T Labs Research, 180 Park Ave., PO Box 971, Florham Park, NJ 07932.*
E-mail: {agilbert, kotidis, muthu, mstrauss}@research.att.com.

Manuscript received 23 May 2002; revised 5 Dec. 2002; accepted 7 Dec. 2002. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 117930.

2 Related Work

Streaming or one-pass algorithms have been studied in different areas. In algorithms, streaming models have been studied in [3], [13], [14], [25], [24], 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 obtaining median, quantiles, and other order statistics [1], [30], [19], correlated aggregate queries [16], mining [15], [9], etc.

Recently, there is an effort to study the general principles behind continuous queries over data streams in the database community (e.g., [4], [29]); this is different from our approach here of supporting ad hoc estimates over the distribution. The problem of summarizing a data stream to provide point and range estimates was first considered in the conference version of this article [22]. Dobra et al. [10] build on the idea of using sketches to answer complex multijoin aggregate queries over multiple data streams. A sketch-based architecture for collecting and processing distributed network measurements is discussed in [20].

Many techniques have been devised in databases for small space approximations in the context of selectivity estimation: sampling techniques [23], [35], histograms [34], wavelet methods [31], [5], etc. Much of this work is for static data sets and they do not provide strict performance guarantees in a streaming 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. Randomized algorithms have been studied in [18] for dynamically maintaining V-Opt histograms and in [26] for multidimensional histogram maintenance.

Recently, the problem of maintaining wavelet transforms as data dynamically changes was considered in [32]; 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). 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* [12]) is in detecting which infrequent values become significantly frequent as data items accumulate, using small amount of space. The authors in [27] propose maintaining a *fixed* set of transform coefficients over time. The authors in [32] 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.

In addition to the wavelet-based methods, the authors in [27] attempted to maintain the significant Discrete Cosine transform values over time. In [17], nonadaptive sampling is used for finding a small-space Fourier representation, such that the sum-squares-error of the representation is within the factor $(1+\epsilon)$ of best possible error.

3 Data Streams and Query Processing

3.1 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,[2] and the function maps it to nonnegative integers.

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.[3] In the *aggregate model*, the items that 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. 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:

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

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.

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

3. The flow of sales through a cash register in a store generates a stream in this model. More generally, one may consider transactions that also "subtract" from the underlying data distribution.

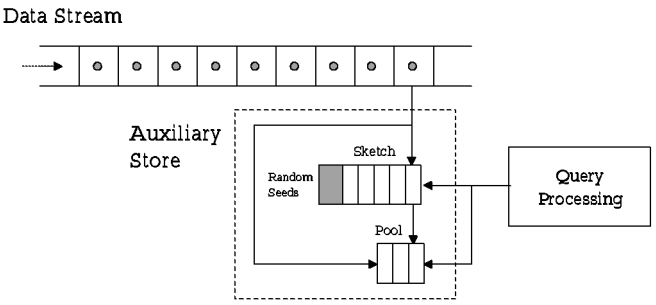

Fig. 1. Stream processing.

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 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.

3.2 Stream Processing Model

Now, we focus on 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 Fig. 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 to keep both as small as possible.

3.3 Aggregate Queries on Streams

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)[4] 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 a particular telephone number involved?* This is a typical "point" query on the signal. *How many total minutes of calls were handled by a telephone exchange which is given by a particular npa-nxx combination?* This is a typical "range" query.

Our methods apply to both 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 is and a maximum bound for the $a(i)$s. Our techniques can be easily extended to the case when neither is known in advance.

4 Background on Wavelet Transforms

Wavelet transforms (like Discrete Cosine and Fourier transforms) are special mathematical transforms that attempt to capture the trends 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 [31] for similar background. Consider the signal of length[5] $N = 8$ 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 $\{1/\sqrt{2}, 1/\sqrt{2}\}$ 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 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.

4. 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.

5. 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.

TABLE 1
Haar Wavelet Decomposition of Array $A = [1, 3, 5, 11, 12, 13, 0, 1]$

Level 3	1	3	5	11	12	13	0	1
Level 2	2.8284	11.3137	17.6777	0.7071	**1.4142**	**4.2426**	**0.7071**	**0.7071**
Level 1	10.0000	13.0000	**6.0000**	**-12.0000**				
Level 0	16.2635	**2.1213**						

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 with the bottommost level being 0 and the topmost (the array) 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. These ranges where $\phi_{j,k}(l) = 1$ are known as *dyadic* intervals in the literature.

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 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.

Formally, we refer to the coefficients ($N - 1$ "differences" and one "average"), as *wavelet basis coefficients* and denote them by w_ℓ, so

$$\{w_\ell : \ell = 0, 1, \ldots, N - 1\} = \{c_{0,0}\} \cup \{d_{j,k}\}.$$

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. For later use, we define the *support* of $\psi_{j,k}$ to be all ls such that $\psi_{j,k}[l] = 1$; support of v is the entire domain.

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-test 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., $\sum |a_i|^2 = \sum |w_\ell|^2$. To measure the goodness of a representation, we define the *pseudoenergy* of a representation R for signal a to be $\|a\|_2^2 - \|a - R\|_2^2$. We use pseudoenergy instead of energy $\|R\|_2^2$ to account for inexact coefficients. Note that if the coefficients of R are optimal for the choice of vectors in R, then the pseudoenergy of R equals the energy of R and maximizing the energy of R amounts to choosing vectors in R to minimize the sse of R. For our representations, however, the coefficients may be suboptimal. One can then raise the energy of R by enlarging a coefficient, which does not lower the sse. We use the above definition of pseudoenergy so that maximizing the pseudoenergy corresponds directly to minimizing the sse $\|a - R\|_2^2$ and the energy and pseudoenergy coincide for exact coefficients.

4.1 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 rapid-decaying behavior by which increasing B beyond a small "threshold" does not significantly decrease the sum-squares-error. As an example, Fig. 2 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 [7], as well as approximate query answering for point and range queries [31], [5], [21]. 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 has 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.

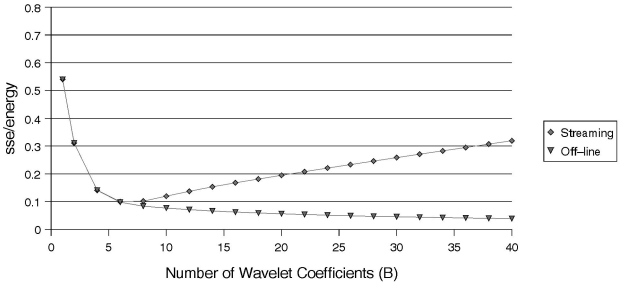

Fig. 2. Ratio sse/energy v.s. B for day 0.

5 Some Foundational Issues

Let us recall that our goal is to compute the highest (best) B-term approximation to a signal of domain size N. 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 4. Recall also that in the ordered aggregate model, we get the signal values specified as $(i, a(i))$ in the increasing order of is. 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_is and d_is 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 [12], [6], [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.

Theorem 2. *Any streaming algorithm that correctly calculates the* second *largest wavelet basis coefficient (which is the largest wavelet coefficient) of the signal rendered by unordered, cash register streaming data uses at least $N/\log^{O(1)}(N)$ space. In fact, this holds if*

1. *The algorithm need only estimate the quantity up to the factor 2 and not necessarily compute it exactly,*
2. *The algorithm need only identify a wavelet basis coefficient (other than the largest) whose magnitude is at least half the magnitude of the second largest one and not necessarily approximate the value of any particular coefficient, and*
3. *The stream is at most $O(N)$ in size and no larger stream is needed to force this lower bound.*

Additionally, a lower bound of $\Omega(N/polylog(N))$ holds for the unordered aggregate stream rendition of the signal.

Proof. We reduce the problem of approximating the value of the largest single coefficient from the following communication complexity problem, MULTIPARTY SET DISJOINTNESS [3]: There are s players, each of whom holds a set of size t from a universe of size approximately $2st$. The sets are promised either to be disjoint or to intersect uniquely—that is, for some i, each set contains i and, for all $j \neq i$, zero or one of the sets contains j. The players must determine which case is in effect. By [3], this requires communication $\Omega(t/s^3)$ in the private coin randomized model. We will use the public coin model—there's a (long) list of random numbers to which all the players have access at no cost. Simulating a public-coin protocol by a private coin protocol requires $\log(N) + \log(1/\delta)$ additional communication bits, where δ (here a constant) is the change in success probability, so the problem is hard in the public coin model, too.

In our case, there are $s \leq O(\log(N))$ players, each of whom holds a t-set from a universe of size $N = 2st$, such that the sets are either pairwise disjoint or have a common single intersection point. Each player regards his share of the MULTIPARTY SET DISJOINTNESS instance (an array of zeros and ones) as a signal on the odd-indexed points $1, 3, \ldots, N-1$; the even-indexed signal values are all zero. Using common randomness, each signal position is negated, independently, with probability $1/2$. Thus, the combined signal has a single peak of height $\pm s$, say at position i, in the intersecting case, and, in that case, the wavelet coefficient $(a_i - a_{i-1})/\sqrt{2}$ has the value $\pm s/\sqrt{2}$. All other signal points in the intersecting case and all signal points in the disjoint case, are $0, 1$, or -1. (We will make all the signal points nonnegative below.) In the disjoint case, all wavelet coefficients at level $j = 1$, consisting of $1/\sqrt{2}$ times the difference of adjacent signal points, have the value zero or $\pm 1/\sqrt{2}$.

We now show that no wavelet basis coefficient (in lower levels) has magnitude greater than $s/2\sqrt{2}$ in the disjoint case. Coefficients at level j are the result of a random walk of length at most $2^{j/2}$ with equally-likely steps of size $\pm 2^{-j/2}$. The expected absolute value of the result is 1, and, with probability at least $1 - 1/N$, the absolute value of each result is at most $\log(N)$, which we

can assume is less than $s/2\sqrt{2}$. Thus, with constant probability, no such coefficient is larger in magnitude than $s/2$. (Note that, if in the intersecting case, some coefficient other than our target is larger than s, we would still be able to distinguish the intersecting and disjoint cases.)

This implies a lower bound of $t/s^4 = n/\log^5(N)$ space for a signal of N possible distinct types and $\Theta(N)$ stream items.

If we restrict attention to nonnegative signals, the problem is still hard since one can reduce the general problem to the nonnegative promise problem. One of the players simply adds the constant vector with $2s$ in each component to the signal constructed above. The largest wavelet basis coefficient will be the average $c_{0,0}$, which takes the value $2s \pm s/2\sqrt{2} \geq s$. The second largest wavelet basis coefficient, which is the largest wavelet coefficient, will be the largest wavelet basis coefficient in the original signal that consists of both positive and negative coefficients. In the cash register model, the constant s signal requires just N additional updates, so the total number of updates (the stream length) is $\Theta(N)$.

Next, consider finding the identity of a near-maximal coefficient. Once we know the identity, we could make a second pass and determine its value in $O(\log(N))$ space, thereby getting a constant-factor approximation to the maximum value using roughly the same communication.

Finally, the proof for the unordered aggregate model is quite similar. □

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.

6 Our Data Streaming Algorithms

6.1 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 the domain of size N, the sketch is of size $\log^{O(1)}(N)$. 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.

6.2 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.

6.2.1 Sketches

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 random vector of ± 1-valued random variables. This is the standard random projection approach found, e.g., in [3]. A *sketch* of the signal is $O(\log(N/\delta)/\epsilon^2)$ independent atomic sketches, each with a different random vector r. (Below, we will substitute other values for ϵ and/or δ.)

Since this sketch size is rather small compared to the signal size, we explicitly store the sketch in the auxiliary store. As the 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^js. Notice that the r^js are of length N each, and, above, we argued that space N is prohibitive. The idea in [25] (following [3]) is that each r^j can be generated by a pseudorandom number generator from a seed s^j of size $\log^{O(1)}(N)$, which is stored explicitly. The generator G takes in s^j and i and outputs $r_i^j = G(s^j, i)$, quickly. Such random vectors are easy to generate, as shown in [25]. For a provably correct algorithm, any four random variables need to be independent; in practice, for some data, other pseudorandom number generators may also work.

We adopt this approach, but our requirements on the random vectors r^js 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 nonzero entries. Explicitly generating r_i^j for each i with nonzero wavelet basis vector component will thus prove time consuming. We need to be able to compute $\langle r^j, \psi \rangle$ quickly. With the exception of the single "average," this inner product is actually the difference of two rangesums of the random variables r_i^j computed over two consecutive dyadic ranges over i. Henceforth, we drop the superscript j in r^j. Our generator is described next.

6.2.2 The Reed-Muller-based Generator

We show how to generate r^js to meet our requirements above. The method we use, at the suggestion of Eric Rains and Neil Sloane, is based on second-order Reed-Muller codes. We assume that N is a power of 2, by rounding the actual value of N up to the next power of 2. Thus, for each j, we need a data structure for a family of N random variables with values of ± 1, with the following properties:

- Any four random variables in the family are independent.
- For any range $[\ell, r)$, $\sum_{\ell \le i < r} r_i$ can be computed quickly, in time $\log(N/\delta)(B/(\epsilon\eta))^{O(1)}$, though we will achieve $\log^{O(1)}(N)$. In particular, r_i can be computed quickly.
- The data structure has size at most

$$\log(N/\delta)(B/(\epsilon\eta))^{O(1)},$$

  though, again, we will achieve $\log^{O(1)}(N)$.

As discussed above, we will use $\log(N/\delta)(B/(\epsilon\eta))^{O(1)}$ independent copies of the data structure, indexed by j.

The data structure stores, as a seed, a bit string s of length $m = 1 + \log(N) + \binom{\log(N)}{2}$. Then, r_i is given by $(-1)^{\langle s, M_i \rangle}$, where the dot product is modulo 2, and M_i is the ith column of the $m \times N$ 2^{nd} order Reed-Muller parity-check matrix, M, which we describe below. Thus, the size of the data structure is the size of the seed, $O(\log^2(N))$. It is shown in [28] (in a different context) that the family of random variables satisfies the other properties above.

We now describe the generator in more detail. We remark that the details are included here for convenience; much of it can be derived from [28], but requires specializing it to our context.

The matrix M is defined as follows: Pick $\log N$ symbols, $a, b, c, \ldots$. The N columns are indexed by subsets of the symbols, in a reverse lexicographic order (see below). The m rows of M and the m bits of s are indexed by subsets of the symbols of size 0, 1, or 2. The entry in row r and column c is 1 if $r \subseteq c$, and 0 otherwise.

For example, for $N = 16$, the matrix is given below. The row indices are listed; the column indices can be read from the four rows with singleton indices.

$\emptyset$:	1111	1111	1111	1111
$\{d\}$	0000	0000	1111	1111
$\{c\}$	0000	1111	0000	1111
$\{b\}$	0011	0011	0011	0011
$\{a\}$	0101	0101	0101	0101
$\{c,d\}$	0000	0000	0000	1111
$\{b,d\}$	0000	0000	0011	0011
$\{a,d\}$	0000	0000	0101	0101
$\{b,c\}$	0000	0011	0000	0011
$\{a,c\}$	0000	0101	0000	0101
$\{a,b\}$	0001	0001	0001	0001

To implement our algorithm, one needs to know how to compute $\sum_{\ell \le i < r} r_i$ quickly; we sketch the algorithm here, directing the reader to [28] for omitted details. We first give an overview; after that, we provide enough details to implement.

Overview. We will show below that, using symmetry, it suffices to sum $\sum_{0 \le i < N} r_i$, i.e., to assume that the range contains all the random variables. We can identify a seed s with a function $s(i) = r_i$, that takes a number $i \in [0, N)$ and outputs a pseudorandom bit. We also identify a seed s with the subset of rows of M corresponding to the 1s in s; then, r_i is the ith position in the mod 2 sum of these rows. Assuming for the moment that the bit of s indexed by $\emptyset$ is 0, we also identify s with a particular $\log(N) \times \log(N)$ matrix Q, a "quadratic form," such that

- Q is zero below the diagonal,
- Entry Q_{jj} equals the bit of s indexed by the singleton set of the jth symbol, and
- Entry Q_{jk}, $j < k$, equals the bit of s indexed by the doubleton set of the jth and kth symbols.

As we show below, r_i is given by the mod 2 matrix product iQi^T, where we regard i as the vector of the binary expansion of the number i. Our goal is to compute $\sum_i r_i$, where the sum is over the integers. For certain seeds s, called *easy*, $\sum_i r_i$ is straightforward to compute. For example, in the example above, if the $\{a, d\}$ and $\{b, d\}$ bits of s are zero, then r has the form $\rho_1\rho_2\rho_3\rho_4$, where the ρs are equal or opposite, depending on the $\{c, d\}$, $\{c\}$, and $\{d\}$ bits. In any of these cases, we can compute $\sum_{0 \le i < N/4}(\rho_1)_i = \sum_{0 \le i < N/4} r_i$ recursively and use this value to compute $\sum_{0 \le i < N} r_i$ immediately. More generally, we require that, in an easy seed, the only nonzero doubleton-indexed bits are $\{a, b\}, \{c, d\}, \{e, f\}$, etc.

Since our goal is to compute $\sum_i r_i = \sum_i iQi^T$, we get the same sum if we permute the r_is. Below, using elementary matrix operations, we will find a $\log(N) \times \log(N)$ matrix R such that

- R is invertible so that $i \mapsto iR$ is a permutation. It follows that

$$\sum_i iQi^t = \sum_i (iR)Q(iR)^t = \sum_i i(RQR^T)i^T.$$

- (A matrix equivalent to) RQR^T is identified with a known easy seed, s'.

We then compute the sum of nonzero bits for the easy seed s' instead of a general seed s. This completes the overview. Now, we will provide details.

Formal Description. We assume that the range $[\ell, r)$ is dyadic since any range can be partitioned into $O(\log(N))$ dyadic subranges. Similarly, it suffices to assume $[\ell, r)$ is the entire range $[0, N)$ since other dyadic ranges are equivalent to instances of our problem for smaller N. For example, consider columns $[4, 8)$ of the second-order Reed-Muller matrix for $N = 16$. After deleting useless zero rows (which can be realized by deleting corresponding bits in s) and removing repeated rows (which can be realized by replacing the corresponding bits in s by a single bit equal to their mod 2 sum), we end up with the second-order Reed-Muller matrix for $N = 4$.

Note that $\sum_{0\le i<N}(-1)^{\langle s,M_i\rangle} = 2\sum_{0\le i<N}\langle s, M_i\rangle - N$. We briefly summarize the technique implied in [28] for computing $\sum_{0\le i<N}\langle s, M_i\rangle$.

We can equivalently regard $\langle s, M_i\rangle$ as $iQ(s)i^t \bmod 2 + Li^t + C$ modulo 2, where i is a vector of $\log(N)$ 0's and 1's mapping the subset of symbols that we identify with index i (it turns out that the vector is actually just the binary expansion of the value i), $Q(s)$ is a $\log(N) \times \log(N)$ matrix whose (α, β) entry equals the $\{\alpha, \beta\}$ entry of s, where α and β are symbols among $a, b, c, \ldots$, and $\alpha \le \beta$, L is a vector whose αth entry equals the $\{\alpha\}$th entry of s, and C is a constant equal to the $\emptyset$th bit of s.

Corresponding to Q is a matrix $B = Q + Q^t$, that is symmetric and has zero diagonal. We next find an invertible matrix R such that RBR^t has the form

$$B' = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & \\ 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 0 & \\ 0 & 0 & 1 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & \ddots \\ & & & & \ddots & \end{pmatrix},$$

where the diagonals of alternating 1s have equal length, but may terminate before reaching the other end. Note that any such matrix is identified with an easy seed. To find R, first use elementary row and column operations to put all the nonzero rows and columns of B into the upper left-hand corner. We can now ignore all the zero rows and columns. Next, recursively make all but the bottom row and right column into the desired form, by using a matrix R of the form

$$\begin{pmatrix} R' & 0 \\ 0 & 1 \end{pmatrix}.$$

Thus, we are left with a matrix of the form

$$\begin{pmatrix} 0 & 1 & \cdots & 0 & 0 & 0 & a_1 \\ 1 & 0 & \cdots & 0 & 0 & 0 & a_2 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 & 0 & a_{2t-1} \\ 0 & 0 & \cdots & 1 & 0 & 0 & a_{2t} \\ 0 & 0 & \cdots & 0 & 0 & 0 & a_{2t+1} \\ a_1 & a_2 & \cdots & a_{2t-1} & a_{2t} & a_{2t+1} & 0 \end{pmatrix}$$

or

$$\begin{pmatrix} 0 & 1 & \cdots & 0 & 0 & a_1 \\ 1 & 0 & \cdots & 0 & 0 & a_2 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 & a_{2t-1} \\ 0 & 0 & \cdots & 1 & 0 & a_{2t} \\ a_1 & a_2 & \cdots & a_{2t-1} & a_{2t} & 0 \end{pmatrix}$$

depending on whether the rank of B is even or odd. Then, use

$$R = \begin{pmatrix} I & 0 \\ r & 1 \end{pmatrix},$$

where I is an identity matrix and r is a row of the form $(a_2, a_1, a_4, a_3, \ldots a_{2t}a_{2t-1})$ or $(a_2, a_1, a_4, a_3, \ldots a_{2t}a_{2t-1}, 0)$. Note that these operations on a $\log(N) \times \log(N)$ matrix take time $\log^{O(1)}(N)$.

Thus, we have $RBR^t = B'$, of the desired form. Since R is invertible, it follows that $\sum_i iQi^t = \sum_i (iR)Q(iR)^t$ since the sums are over the same set. This is $\sum_i i(RQR^t)i^t$, which equals $\sum_i i\mathrm{UT}((RQR^t) + (RQR^t)^t)i^t = \sum_i i\mathrm{UT}(B')i^t$, where UT denotes the upper triangle, since, for any matrix A,

$$xAx^t = x\mathrm{UT}(A + A^t)x^t,$$

because, working modulo 2, each side is the parity of the number of pairs (j, k) such that $x_j = x_k = 1$ and $A_{jk} + A_{kj} = 1$. Similarly,

$$\begin{aligned} \sum_i iQi^t + Li^t + C &= \sum_i (iR)Q(iR)^t + L(iR)^t + C \\ &= \sum_i i\mathrm{UT}(B')i^t + i(RL^t) + C \\ &= \sum_i i(\mathrm{UT}(B') + \mathrm{diag}(RL^t))i^t + C, \end{aligned}$$

since $x^2 = x$ so $Li^t = i\mathrm{diag}(L)i^t$, modulo 2. We may assume that $C = 0$, since, if $C = 1$, the sum becomes

$$N - \sum_i i(\mathrm{UT}(B') + \mathrm{diag}(RL^t))i^t.$$

(Recall that the summation is over the integers, but all the matrix arithmetic is performed modulo 2.)

Finally, the matrix $(\mathrm{UT}(B') + \mathrm{diag}(RL^t))$ is in a form simple enough that we can perform the desired sum recursively. That is, $(\mathrm{UT}(B') + \mathrm{diag}(RL^t))$ is of the form

$$\begin{pmatrix} a_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_3 & 1 & 0 & 0 \\ 0 & 0 & 0 & a_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_5 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

where the main diagonal is arbitrary and the diagonal above the main diagonal alternates 0s and 1s at first and may eventually become all zeros. The corresponding seed, defined using the symbols α_k, has 1s only in the positions indexed either by singletons or by doubletons of the form $\{\alpha_{2j-1}, \alpha_{2j}\}$. For example, if we remove the doubleton-indexed rows of M above corresponding to bits of s that are now forced to zero, assuming the alternating diagonal is 1010 rather than, say, 1000, we get

$$M' = \begin{pmatrix} 0000 & 0000 & 1111 & 1111 \\ 0000 & 1111 & 0000 & 1111 \\ 0011 & 0011 & 0011 & 0011 \\ 0101 & 0101 & 0101 & 0101 \\ & & & \\ 0000 & 0000 & 0000 & 1111 \\ 0001 & 0001 & 0001 & 0001 \end{pmatrix}.$$

Observe that, by the structure of M, each row in M' is of the form $\rho_1\rho_2\rho_3\rho_4$, where any two ρ's are either equal or opposite. It follows that the vector sM, the sum of some set of rows, is the concatenation of four pieces, where any two pieces are equal or opposite. One can recursively find the number of 1s in one of these pieces of length $N/4$, then, in constant time, use that value to compute the number of 1s among all N of the random variables.

6.2.3 Estimation of Inner Products

We need a technical primitive, namely, estimating the inner product of two vectors given their sketches. Sketches were first introduced to estimate norms of vectors, e.g., in [3]. In our experiments, we used an approximation for $\langle a, b\rangle$ based on the identity

$$\langle a, b\rangle = \|a\|\|b\|\left(1 - \frac{\left\|a/\|a\| - b/\|b\|\right\|^2}{2}\right).$$

An anonymous referee pointed us to a previous, simpler technique, proved in [2]: The product of a sketch for a and a sketch for b is an estimate for $\langle a, b\rangle$ with similar guarantees. Either way, the following holds (see [2], [3] for a proof).

Lemma 3.

1. *Let X be the $O(\log(1/\delta))$-wise median of $O(1/\epsilon^2)$-wise means of independent copies of*

$$\left(\sum_i a_i r_i^j\right)\left(\sum_i b_i r_i^j\right).$$

Then, we have $|X - \langle a, b\rangle| \le \epsilon\|a\|\|b\|$ with probability $1 - \delta$.

2. *Let Y be the $O(\log(1/\delta))$-wise median of $O(1/\epsilon^2)$-wise means of independent copies of*

$$\left(\sum_i a_i r_i^j\right)^2.$$

Then, we have $\left|Y - \|a\|^2\right| \le \epsilon\|a\|^2$ with probability $1 - \delta$.

Below, we will substitute other values for ϵ and δ in this lemma.

6.2.4 Answering Queries from the Sketch

We consider two models of query processing: *batch* and *online*. In the batch model, queries may be posed only at periodic intervals, e.g., at the end of the day. We do not need to answer queries in midstream so we can perform some time-consuming additional processing during the batch process and amortize this cost over the entire input stream and the collection of queries.

In the online model, queries may be posed at any time during the stream processing. Thus, we must be able to respond rapidly to queries in this case. We present two heuristic methods: one using wavelet coefficients that pertain only to the query point and the second that estimates point values directly.

6.2.5 Batch Query Processing

To answer queries in a batch, we first build a B-term wavelet representation to the entire signal using the techniques in Section 6.2. To do this, we exhaustively estimate each wavelet coefficient as described and select up to B of the largest coefficients (excluding those whose square is less than $\eta\epsilon/B\|a\|_2^2$). We use these estimates as coefficients in an almost B-term approximation to the signal. To answer a query in time $O(B + \log N)$, we determine the $O(\log N)$ wavelets whose supports intersect the query point and sum over those in our B-term approximation. Note that this procedure also works for range queries. This procedure provides provable accuracy guarantees over all point queries.

Theorem 4. *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 pseudoenergy at least $(1 - \epsilon)\eta\|a\|_2^2$. If there is no B-term representation with energy $(1 - \epsilon)\eta\|a\|_2^2$, A reports "no good representation." In any case, A uses*

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

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

Proof. First, estimate $\|a\|$ as $\|a\|_\sim$, with

$$\|a\|_\sim^2 \le \|a\|^2 \le 2\|a\|_\sim^2.$$

Next, estimate each coefficient, $\langle a, \zeta_j\rangle$, as $\langle a, \zeta_j\rangle_\sim$, with

$$\langle a, \zeta_j\rangle_\sim = \langle a, \zeta_j\rangle \pm x_1\|a\|\|\zeta_j\| = \langle a\zeta_j\rangle \pm x_1\|a\|,$$

where x_1 will be determined below. From among the B largest estimates, take those with $\langle a, \zeta_j\rangle_\sim > x_2\|a\|_\sim$, where x_2 will be determined below; denote by $\Lambda_\sim$ the set of chosen indices. Let Λ' be the set of indices of the $|\Lambda_\sim|$ biggest coefficients, and let Λ be the set of indices of the B largest.

Suppose $\langle a, \zeta_{j_1}\rangle_\sim > \langle a, \zeta_{j_2}\rangle_\sim$, but $\langle a, \zeta_{j_1} < \rangle a, \zeta_{j_2}\rangle$. We claim that all four quantities are close in relative value. Specifically,

$$\begin{aligned}\langle a, \zeta_{j_1}\rangle &> \langle a, \zeta_{j_1}\rangle_\sim - x_1\|a\| > x_2\|a\|_\sim - x_1\|a\| \\ &\ge (x_2/\sqrt{2})\|a\| - x_1\|a\| = (x_2/\sqrt{2} - x_1)\|a\|\end{aligned}$$

so $\langle a, \zeta_{j_1}\rangle_\sim = \langle a, \zeta_{j_1}\rangle \pm x_1\|a\| \subseteq \langle a, \zeta_{j_1}\rangle\left(1 \pm \frac{x_1}{x_2/\sqrt{2} - x_1}\right)$. Similarly,

$$\langle a, \zeta_{j_2}\rangle > \langle a, \zeta_{j_1}\rangle > (x_2/\sqrt{2} - x_1)\|a\|$$

and $\langle a, \zeta_{j_2}\rangle_\sim = \langle a, \zeta_{j_2}\rangle \pm x_1\|a\| \subseteq \langle a, \zeta_{j_2}\rangle\left(1 \pm \frac{x_1}{x_2/\sqrt{2} - x_1}\right)$.

It follows that

$$\begin{aligned}\langle a,\zeta_{j_2}\rangle_\sim &< \langle a,\zeta_{j_1}\rangle_\sim \\ &< \langle a,\zeta_{j_2}\rangle\cdot\left(1+\frac{x_1}{x_2/\sqrt{2}-x_1}\right)\cdot\left(1-\frac{x_1}{x_2/\sqrt{2}-x_1}\right)^{-1} \\ &< \langle a,\zeta_{j_2}\rangle_\sim\cdot\left(1+3\frac{x_1}{x_2/\sqrt{2}-x_1}\right),\end{aligned}$$

provided $\frac{x_1}{x_2/\sqrt{2}-x_1}$ is sufficiently small.

It follows that the best representation over the terms in $\Lambda_\sim$ has energy within $B\left(3\frac{x_1}{x_2/\sqrt{2}-x_1}\right)^2\|a\|^2$ of the energy of the best representation over Λ'. If $|\Lambda'|=B$, then we have shown that the energy of the best representation over the coefficients chosen is within $B\left(3\frac{x_1}{x_2/\sqrt{2}-x_1}\right)^2\|a\|^2$ of the energy of the best B-term representation. Otherwise, consider any coefficient, with index j, of magnitude at least $(x_2+x_1)\|a\|$. Then, since we have not chosen B coefficients, the jth coefficient must be among those chosen since $\langle a,\zeta_j\rangle_\sim \geq x_2\|a\| \geq x_2\|a\|_\sim$. Thus, the atmost-$B$ of the top coefficients *not* chosen have combined energy at most $B(x_2+x_1)^2\|a\|^2$. It follows that, if there is a B-term representation with energy $\eta\|a\|^2$, then the best representation over the chosen coefficients has energy at least

$$\left(1-B\left(3\frac{x_1}{x_2/\sqrt{2}-x_1}\right)^2\right)\eta\|a\|^2 - B(x_2+x_1)^2\|a\|^2 \leq (1-\epsilon/3)\eta\|a\|^2,$$

if $x_2=c_2\sqrt{\epsilon\eta/B})$ and $x_1=c_1x_2\sqrt{\epsilon/B}=c_1c_2\epsilon\sqrt{\eta}/B$ for some constants c_1 and c_2. The cost, which depends on x_1, but not x_2, is $1/x_1^2$, i.e., $O(B^2/(\epsilon^2\eta))$.

We cannot recover the best coefficient values. However, any coefficient we estimate has pseudoenergy within $x_1^2\|a\|^2$ less than the best value for that coefficient. So, the representation we recover has pseudoenergy within $Bx_1^2\|a\|^2 \leq B(x_2+x_1)^2\|a\|^2 \leq (1-\epsilon/3)\eta\|a\|^2$ of best possible over $\Lambda_\sim$. So, if there is a representation with energy $\eta\|a\|^2$, we find a representation R with pseudoenergy at least $(1-2\epsilon/3)\eta\|a\|^2$. (To get the proof of the theorem, we can work with ϵ' so that $2\epsilon'/3$ becomes the ϵ and the theorem follows.)

Now, suppose there is no B-term representation with energy $(1-\epsilon)\eta\|a\|^2$. The above procedure may produce a representation, R, but we should not output it. We next show that we can test any representation R produced as above to see whether it is good enough.

Estimate $\|a\|$ as $\|a\|_\sim$ with $\|a\|^2 \leq \|a\|_\sim^2 \leq (1+y_1)\|a\|^2$, for y_1 to be determined below, and estimate $\|a-R\|$ as $\|a-R\|_\sim$ with $\|a-R\|^2 \leq \|a-R\|_\sim^2 \leq (1+y_2)\|a-R\|^2$ for y_2 to be determined below. For y_3 to be determined below, if $\|a\|_\sim^2 - \|a-R\|_\sim^2 > y_3\|a\|_\sim^2$, then output R; otherwise, output "no good representation."

Suppose there is an R with $\|a\|^2-\|a-R\|^2>\eta\|a\|^2$. Then, we find a representation with

$$\|a\|^2-\|a-R\|^2>(1-2\epsilon/3)\eta\|a\|^2,$$

i.e., $\|a-R\|^2<(1-\eta+2\eta\epsilon/3)\|a\|^2$. Then,

$$\|a-R\|_\sim^2 \leq (1+y_2)(1-\eta+2\eta\epsilon/3)\|a\|_\sim^2,$$

so

$$\|a\|_\sim^2-\|a-R\|_\sim^2 \geq (-y_2+\eta(1-2\epsilon/3) - y_2\eta(1-2\epsilon/3))\|a\|_\sim^2.$$

On the other hand, if there is no R with

$$\|a\|^2-\|a-R\|^2>(1-\epsilon)\eta\|a\|^2,$$

then, for any possible R we find,

$$\|a\|^2-\|a-R\|^2\leq(1-\epsilon)\eta\|a\|^2,$$

so that

$$\begin{aligned}\|a\|_\sim^2-\|a-R\|_\sim^2 &\leq (1-\epsilon)\eta\|a\|^2+y_1\|a\|^2 \\ &\leq ((1-\epsilon)\eta+y_1)\|a\|_\sim^2.\end{aligned}$$

Thus, we need

$$(1-\epsilon)\eta+y_1\leq y_3\leq -y_2+\eta(1-2\epsilon/3)-y_2\eta(1-2\epsilon/3),$$

which can be satisfied for $y_2=\epsilon\eta/6$ and y_1 also linear in $\epsilon\eta$. It follows that the space overhead and processing times are quadratic in $1/(\epsilon\eta)$. (The value for y_3 does not affect the cost.) □

Note that if there is a representation with pseudoenergy at least $(1-\epsilon)\eta\|a\|_2^2$, but none with pseudoenergy at least $\eta\|a\|_2^2$, then our algorithm may arbitrarily answer "no good representation" or it may output a representation with pseudoenergy at least $(1-\epsilon)\eta\|a\|_2^2$. Either behavior is semantically correct.

6.2.6 Online Query Processing

To answer point queries immediately and directly from the sketch, we can choose from two heuristic procedures: *direct point* or *direct wavelet* estimates.

For a point query i, we associate it with the vector e_i, consisting of a 1 in position i and zeros elsewhere. An atomic sketch of e_i using the random variable r^j is simply r_i^j. To estimate $a_i=\langle a,e_i\rangle$, we compute the sketches of e_i and of a and use Lemma 3 with probability δ/N and distortion $\eta\epsilon$. If $|a_i|\geq\eta\|a\|$, then our estimate will be $a_i(1\pm\epsilon)$ except with probability δ/N, in space $\log(N/\delta)/(\eta\epsilon)^2$. 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.

We can also answer a point query i relatively immediately by estimating directly the $O(\log N)$ wavelet coefficients corresponding to those wavelets whose supports intersect i. That is, we perform a direct wavelet estimate (as compared to a direct point estimate). We use the techniques described above and Lemma 3 to estimate the large wavelet coefficients.

6.2.7 Details and Discussion

One drawback to finding the best B-term approximation for batch query processing is that it takes time $O(N\log N)$. This

time 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 so that it contains the highest B coefficient estimates over all. In this way, we amortize the cost of computing the estimates over 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.

While the batch mode provably gives a near best B-term representation, it does so under the assumption that we estimate all point queries, each equally likely. This, however, is not always a valid assumption. Often, we care more about specific queries than others. In this case, direct estimates can be more accurate, but only if the point values contribute significantly to the total energy of the signal. We call these values "heavy hitters" and show in Section 7.2 that direct estimates experimentally perform better than batch estimates on heavy hitters. See this section also for a comparison of direct point versus direct wavelet estimates of heavy hitters.

We also note experimentally that we can boost the performance of both direct methods for approximating heavy hitters using *adaptive greedy pursuit*. That is, we obtain a very accurate estimate for the first heavy hitter (the "heaviest"), we subtract this estimate from the sketch, use the new sketch to estimate the second heavy hitter, etc. Because we subtract the estimate for the first heavy hitter from the sketch and because sketches are linear, the resulting sketch is a good estimate for the sketch of the residual distribution in which the second heavy hitter is the peak value. This increases our chances that we will get an accurate estimate of the second heavy hitter. In this fashion, we estimate the first k heavy hitters, one after the other. Each estimate is subtracted from the sketch, increasing the likelihood that we estimate accurately the remaining heavy hitters. Of course, such a procedure does not work in our favor forever. Each estimate contains a small amount of error that is also subtracted from the sketch. After many iterations, these errors will eventually overwhelm the benefits.

7 Experiments

For the experiments, we used traces from AT&T's call detail data for a period of one week. The data set describes a certain type of long-distance calls, aggregated at the npa-nxx level. The stream is in the unordered cash-register format and contains 511,033,795 values. The npa-nxx value corresponds to the originating number. We present an analysis of the batch method of answering queries (including several heuristics presented in the previous section) and a comparison of the online methods (both direct point and wavelet estimates). We further explore the improvements we obtain by using adaptive greedy pursuit.

7.1 Batch Query Processing

In the first experiment, we used the data for Day 0 (45M records) and computed offline 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 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. At the end of the day, we used the sketch to compute a highest-B approximate set of wavelet coefficients for B between one and 40. We then reconstructed the signal from both highest-B sets (varying B) and computed the point-wise sum-squares-error (sse) of each approximation; that is, the cumulative sse of all point-queries in the npa-nxx domain. Fig. 2 plots the sse of both representations over the energy of the signal, varying B. The seven largest wavelet coefficients are accurately picked by this sketch as they contain (cumulatively) roughly 91 percent of the energy. For $B > 7$, additional wavelet coefficients contain too little information to be reliably identified by the sketch. We can see this in the ratio for the offline case, the error flattens out around seven coefficients.

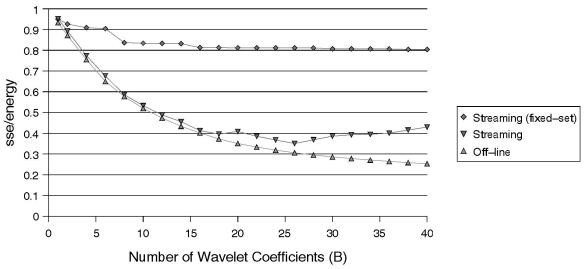

Fig. 3. Ratio sse/energy versus B at the end of day 6.

Because it may be prohibitively expensive to compute all N wavelet coefficients from the sketch and then determine which are the B largest,[6] we perform two experiments to explore the accuracy of possible improvements. We compare using coefficients from a sample of the data and updating coefficients in the background as we stream over the data.

In Fig. 3, we used data from all seven days. We compare the set of B largest coefficients obtained from the sketch with 1) the largest B selection obtained from an offline algorithm (as before) and 2) a static largest 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 [31]. The latter method is denoted as *Streaming (fixed-set)*. As in the first experiment, 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 selection of the B coefficients is fixed a priori.[7]

In a last 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). Fig. 4 shows

6. For this data set, computing a single wavelet coefficient from the sketch takes about 22msecs in a 667Mhz Pentium III PC.

7. To verify that the static selection of coefficients was not hindered by a poor choice in the initial data (Day 0), we reran the experiment starting with Day 1 and found similar results.

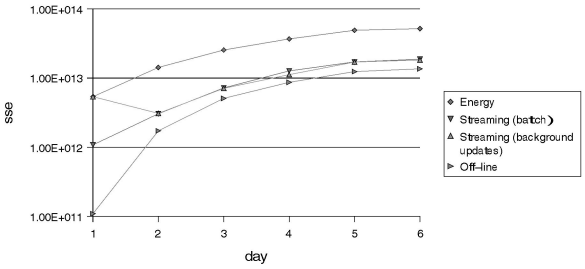

Fig. 4. Sse at the end of each day, starting from day 1.

	Sketch Size		
Method	1958	5434	12232
Direct Point Est.	36.50	5.94	0.06
Direct Wavelet Est.	43.86	7.25	0.06
top-10 Wavelets	0.19	0.09	0.08
top-40 Wavelets	0.59	0.21	0.07

Fig. 5. Comparison (sse/energy) of top-B wavelets against direct estimates.

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 seven times during the six 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.

7.2 Online Query Processing

In Fig. 5, we summarize an experiment using Day 0 of the phone data and compute the sse over the energy of the array for direct point/wavelet and for top-B wavelet (as in the batch process) estimates over all point queries. For the two smaller sketch sizes, the top-B wavelet-based computation is substantially better, in terms of sse, by a factor of 180 and 66, respectively. This is because direct estimation from the sketch results in very gross estimates for all but the largest values of the array, the "heavy hitters." We can also see that adding more than 10 wavelet coefficients in the representation increases the error. This agrees with the observation we made in Fig. 2. These 10 largest wavelet coefficients are the heavy hitters in the wavelet domain and can be estimated reliably by the sketch. The last column of the table shows a large increase in the accuracy of the direct estimates for the larger sketch. We point out though that this sketch size is about 20 percent of the size of the array and not likely to be of real use in practice.

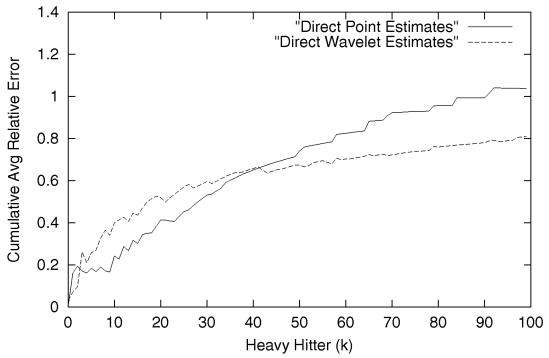

Fig. 6. Direct estimates for the top 100 heavy hitters.

In Table 2, we take a closer look to the estimation of the heavy hitters for the medium sized sketch. We compute the *relative error* $|a[i] - \hat{a}[i]|/a[i]$ for the five largest values of a. The picture now is completely different compared to the previous run. Approximating the heavy hitters using the top-B ($B = 10, 40$) method is very unreliable. This is because the largest B wavelets capture the *overall* distribution, not the peak values. As a result, they are better over all queries (Table 2), but not as good for the heavy hitters.

In Fig. 6, we compute the *average relative error* for the first k heavy hitters for $1 \leq k \leq 100$; that is, the sum of the relative errors for the first k heavy hitters, divided by k. Notice that, unlike the metric sse/energy, this metric is not weighted by the values of the array; that is, all values contribute the same. Notice that initially the direct point estimates are more accurate (in terms of the average relative error) up to heavy hitter 43, but get progressively worse, while the direct wavelet estimates are not as accurate initially, but do not degrade significantly past heavy hitter 50. After heavy hitter 90, the average relative error of the direct point estimates is over 100 percent.

In Fig. 7, we repeat the experiment using adaptive greedy pursuit as described in Section 6.2.4. Direct point estimates are now more accurate than direct wavelet estimates. The average error for the first 100 heavy hitters is down from 104 percent to 15 percent compared to the plain method! We notice, however, that this technique works well when we have a good idea of what we are looking for, say the heavy hitters. Wavelets, on the other hand, are far more reliable for estimating the whole array or randomly-picked parts thereof.

TABLE 2
Relative Error for Top 10 Heavy Hitters

Heavy hitter	Direct Point	Direct Wavelet	top-10 Wavelets	top-40 Wavelets
1	0.21%	3.09%	3.63%	3.32%
2	31.87%	11.03%	99.63%	43.74%
3	26.05%	14.68%	22.92%	22.92%
4	9.96%	75.58%	99.59%	99.59%
5	12.28%	0.70%	99.58%	99.58%

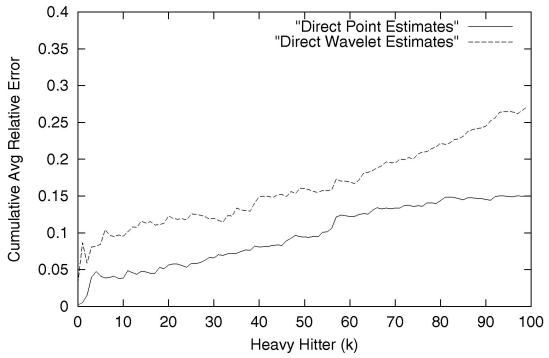

Fig. 7. Direct estimates for the top 100 heavy hitters with adaptive greedy pursuit.

8 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. 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 multidimensions unlike traditional selectivity estimation methods and 2) they have been found to work for data-cube approximations as well [36]. 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 [16]; 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] R. Agrawal and A. Swami, "A One-Pass Space-Ecient Algorithm for Finding Quantiles," *Proc. COMAD,* 1995.

[2] N. Alon, P. Gibbons, Y. Matias, and M. Szegedy, "Tracking Join and Self-Join Sizes in Limited Storage," *Proc. ACM PODS,* pp. 10-20, 1999.

[3] N. Alon, Y. Matias, and M. Szegedy, "The Space Complexity of Approximating the Frequency Moments," *Proc. ACM Symp. Theory of Computing (STOC),* pp. 20-29, 1996.

[4] S. Babu and J. Widom, "Continuous Queries over Data Streams," *SIGMOD Record,* vol. 30, no. 3, pp. 109-120, 2001.

[5] K. Chakrabarti, M.N. Garofalakis, R. Rastogi, and K. Shim, "Approximate Query Processing Using Wavelets," *Proc. VLDB Conf.,* pp. 111-122, 2000.

[6] S. Chaudhuri and L. Gravano, "Evaluating Top-k Selection Queries," *Proc. Very Large Data Bases Conf.,* pp. 397-410, 1999.

[7] K. Chan and A. Fu, "Efficient Time Series Matching by Wavelets," *Proc. Int'l Conf. Data Eng.,* pp. 126-133, 1999.

[8] C. Cortes and D. Pregibon, "Signature-Based Methods for Data Streams," *Data Mining and Knowledge Discovery,* vol. 5, no. 3, pp. 167-182, 2001.

[9] G. Cormode, P. Indyk, N. Koudas, and S. Muthukrishnan, "Fast Mining of Massive Tabular Data via Approximate Distance Computations," *Proc. Int'l Conf. Data Eng.,* 2002.

[10] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi, "Processing Complex Aggregate Queries over Data Stream," *Proc. ACM SIMGOD,* June 2002.

[11] P. Domingos and G. Hulten, "Mining High-Speed Data Streams," *Proc. Int'l Conf. Knowledge Discovery and Data Mining (KDD),* P. Domingos and G. Hulten, eds., pp. 71-80, 2000.

[12] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. Ullman, "Computing Iceberg Queries Efficiently," *Proc. Very Large Data Bases Conf.,* pp. 299-310, 1998

[13] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan, "An Approximate L^1-Difference Algorithm for Massive Data Streams," *Proc. IEEE Symp. Foundations of Computer Science (FOCS),* pp. 501-511, 1999.

[14] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan, "Testing and Spot-Checking of Data Streams," *Proc. ACM-SIAM Symp. Discrete Algorithms (SODA),* pp. 165-174, 2000.

[15] V. Ganti, J. Gehrke, and R. Ramakrishnan, "Mining Very Large Databases," *Computer,* vol. 32, no. 8, pp. 38-45, 1999.

[16] J. Gehrke, F. Korn, and D. Srivastava, "On Computing Correlated Aggregates over Continual Data Streams," *Proc. ACM SIGMOD Conf.,* pp. 13-24, 2001.

[17] A. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss, "Near-Optimal Sparse Fourier Representations via Sampling," *Proc. ACM Symp. Theory of Computing (STOC),* pp. 152-161, 2002.

[18] A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "Fast, Small-Space Algorithms for Approximate Histogram Maintenance," *Proc. ACM Symp. Theory of Computing (STOC),* 2002.

[19] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "How to Summarize the Universe: Dynamic Maintenance of Quantiles," *Proc. Very Large Data Bases Conf.,* pp. 454-465, 2002.

[20] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "QuickSAND: Quick Summary and Analysis of Network Data," DIMACS Technical Report 2001-43, Nov. 2001.

[21] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "Optimal and Approximate Computation of Summary Statistics for Range Aggregates," *Proc. ACM Symp. Principles of Database Systems (PODS),* pp. 227-236, 2001.

[22] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries," *Proc. VLDB Conf.,* pp. 79-88, 2001.

[23] P. Gibbons and Y. Matias, "New Sampling-Based Summary Statistics for Improving Approximate Query Answers," *Proc. ACM SIGMOD Conf.,* pp. 331-342, 1998.

[24] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering Data Streams," *Proc. IEEE Symp. Foundations of Computer Science (FOCS),* pp. 359-366, 2000.

[25] P. Indyk, "Stable Distributions, Pseudorandom Generators, Embeddings and Data Stream Computation," *Proc. IEEE Symp. Foundations of Computer Science (FOCS),* pp. 189-197, 2000.

[26] N. Koudas, S. Guha, P. Indyk, and N. Thaper, "Dynamic Multidimensional Histograms," *Proc. ACM SIGMOD,* pp. 428-439, 2002.

[27] J. Lee, D. Kim, and C. Chung, "Multi-Dimensional Selectivity Estimation Using Compressed Histogram Informantion," *Proc. ACM SIGMOD Conf.,* pp. 205-214, 1999.

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

[29] S. Madden and M.J. Franklin, "Fjording the Stream: An Architecture for Queries over Streaming Sensor Data," *Proc. Int'l Conf. Data Eng.,* 2002.

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

[31] Y. Matias, J.S. Vitter, and M. Wang, "Wavelet-Based Histograms for Selectivity Estimation," *Proc. ACM SIGMOD,* pp. 448-459, 1998.

[32] Y. Matias, J. Vitter, and M. Wang, "Dynamic Maintenance of Wavelet-Based Histograms," *Proc. Very Large Data Bases Conf.,* pp. 101-110, 2000.

[33] N. Nisan, "Pseudorandom Generators for Space-Bounded Computation," *Proc. ACM Symp. Theory of Computing (STOC),* pp. 204-212, 1990.

[34] V. Poosala, Y.E. Ioannidis, P.J. Haas, and E.J. Shekita, "Improved Histograms for Selectivity Estimation of Range Predicates," *Proc. ACM SIGMOD,* pp. 294-305, June 1996.

[35] Y. Wu, D. Agrawal, and A. Abbadi, "Applying the Golden Rule of Sampling for Query Estimation," *Proc. SIGMOD,* pp. 449-460, 2001.
[36] J. Vitter, M. Wang, B. Iyer, "Data Cube Approximation and Histograms via Wavelets," *Proc. Conf. Information and Knowledge Management (CIKM),* pp. 96-104, 1998.

Anna C. Gilbert received the SB degree from the University of Chicago and the PhD degree from Princeton University, both in mathematics. She is a principal technical staff member at AT&T Labs-Research in Florham Park, New Jersey. Her interests include analysis, probability, networking, and algorithms.

Yannis Kotidis received the BSc degree in electrical engineering and computer science from the National Technical University of Athens and the PhD degree in computer science from the University of Maryland. He is a principal technical staff member at AT&T Labs-Research in Florham Park, New Jersey. His interests include data warehousing, approximate query processing and indexing of massive data sets.

S. Muthukrishnan graduated from the Courant Institute of Mathematical Sciences; his thesis work was on probabilistic games and pattern matching algorithms. He was a postdoctoral candidate working on computational biology for a year, on the faculty at the University of Warwick, United Kingdom, has been at Bell Labs, at AT&T Research, and Rutgers University. His research explores fundamental algorithms, as well algorithms applied to databases, networking, compression, scheduling, etc.

Martin J. Strauss received the AB degree from Columbia University and the PhD degree from Rutgers University, both in mathematics, and did a year of postdoctoral research at Iowa State University before joining AT&T. He is a principal technical staff member at AT&T Labs in Florham Park, New Jersey. He has written several articles on algorithms, complexity theory, cryptography, computer security, and other topics. Dr. Strauss is currently interested in algorithms for massive data sets.