Fast Approximate Wavelet Tracking on Streams

Graham Cormode[1], Minos Garofalakis[2], and Dimitris Sacharidis[3]

[1] Bell Labs, Lucent Technologies
cormode@bell-labs.com
[2] Intel Research Berkeley
minos.garofalakis@intel.com
[3] National Technical University of Athens
dsachar@dblab.ntua.gr

Abstract. Recent years have seen growing interest in effective algorithms for summarizing and querying massive, high-speed data streams. Randomized sketch synopses provide accurate approximations for general-purpose summaries of the streaming data distribution (e.g., wavelets). The focus of existing work has typically been on minimizing *space requirements* of the maintained synopsis — however, to effectively support high-speed data-stream analysis, a crucial practical requirement is to also optimize: (1) the *update time* for incorporating a streaming data element in the sketch, and (2) the *query time* for producing an approximate summary (e.g., the top wavelet coefficients) from the sketch. Such time costs must be small enough to cope with rapid stream-arrival rates and the real-time querying requirements of typical streaming applications (e.g., ISP network monitoring). With cheap and plentiful memory, space is often only a secondary concern after query/update time costs.

In this paper, we propose the first fast solution to the problem of tracking wavelet representations of one-dimensional and multi-dimensional data streams, based on a novel stream synopsis, the *Group-Count Sketch (GCS)*. By imposing a hierarchical structure of groups over the data and applying the GCS, our algorithms can quickly recover the most important wavelet coefficients with guaranteed accuracy. A tradeoff between query time and update time is established, by varying the hierarchical structure of groups, allowing the right balance to be found for specific data stream. Experimental analysis confirms this tradeoff, and shows that all our methods significantly outperform previously known methods in terms of both update time and query time, while maintaining a high level of accuracy.

1 Introduction

Driven by the enormous volumes of data communicated over today's Internet, several emerging data-management applications crucially depend on the ability to continuously generate, process, and analyze massive amounts of data in real time. A typical example domain here comprises the class of *continuous event-monitoring systems* deployed in a wide variety of settings, ranging from network-event tracking in large ISPs to transaction-log monitoring in large web-server farms and satellite-based environmental monitoring. For instance, tracking the operation of a nationwide ISP network

Y. Ioannidis et al. (Eds.): EDBT 2006, LNCS 3896, pp. 4–22, 2006.

requires monitoring detailed measurement data from thousands of network elements across several different layers of the network infrastructure. The volume of such monitoring data can easily become overwhelming (in the order of Terabytes per day). To deal effectively with the massive volume and continuous, high-speed nature of data in such environments, the *data streaming* paradigm has proven vital. Unlike conventional database query-processing engines that require several (expensive) passes over a static, archived data image, streaming data-analysis algorithms rely on building concise, approximate (but highly accurate) *synopses* of the input stream(s) in real-time (i.e., in one pass over the streaming data). Such synopses typically require space that is significantly sublinear in the size of the data and can be used to provide *approximate query answers* with guarantees on the quality of the approximation. In many monitoring scenarios, it is neither desirable nor necessary to maintain the data in full; instead, stream synopses can be used to retain enough information for the reliable reconstruction of the key features of the data required in analysis.

The collection of the top (i.e., largest) coefficients in the *wavelet transform* (or, *decomposition*) of an input data vector is one example of such a key feature of the stream. *Wavelets* provide a mathematical tool for the hierarchical decomposition of functions, with a long history of successful applications in signal and image processing [16, 22]. Applying the wavelet transform to a (one- or multi-dimensional) data vector and retaining a select small collection of the largest wavelet coefficient gives a very effective form of lossy data compression. Such *wavelet summaries* provide concise, general-purpose summaries of relational data, and can form the foundation for fast and accurate approximate query processing algorithms (such as approximate selectivity estimates, OLAP range aggregates and approximate join and multi-join queries. Wavelet summaries can also give accurate (one- or multi-dimensional) *histograms* of the underlying data vector at multiple levels of resolution, thus providing valuable primitives for effective data visualization.

Most earlier stream-summarization work focuses on minimizing the *space requirements* for a given level of accuracy (in the resulting approximate wavelet representation) while the data vector is being rendered as a stream of arbitrary point updates. However, while space is an important consideration, it is certainly not the only parameter of interest. To effectively support high-speed data-stream analyses, two additional key parameters of a streaming algorithm are: (1) the *update time* for incorporating a streaming update in the sketch, and (2) the *query time* for producing the approximate summary (e.g., the top wavelet coefficients) from the sketch. Minimizing query and update times is a crucial requirement to cope with rapid stream-arrival rates and the real-time querying needs of modern streaming applications. Furthermore, there are essential tradeoffs between the above three parameters (i.e., space, query time, and update time), and it can be argued that space usage is often the *least* important of these. For instance, consider monitoring a stream of active network connections for the users consuming the most bandwidth (commonly referred to as the "top talkers" or "heavy hitters" [6, 18]). Typical results for this problem give a stream-synopsis space requirement of $O(1/\epsilon)$, meaning that an accuracy of $\epsilon = 0.1\%$ requires only a few thousands of storage locations, i.e., a few Kilobytes, which is of little consequence at all in today's off-the-shelf systems

featuring Gigabytes of main memory[1]. Now, suppose that the network is processing IP packets on average a few hundred bytes in length at rates of hundreds of Mbps; essentially, this implies that the average processing time per packet must much less than one millisecond: an average system throughput of tens to hundreds of thousands of packets per second. Thus, while synopsis space is probably a non-issue in this setting, the times to update and query the synopsis can easily become an insurmountable bottleneck. To scale to such high data speeds, streaming algorithms must guarantee provably small time costs for updating the synopsis in real time. Small query times are also important, requiring near real-time response. (e.g., for detecting and reacting to potential network attacks). In summary, we need fast item processing, fast analysis, and bounded space usage — different scenarios place different emphasis on each parameter but, in general, more attention needs to be paid to the time costs of streaming algorithms.

Our Contributions. The streaming wavelet algorithms of Gilbert et al. [11] guaranteed small space usage, only polylogarithmic in the size of the vector. Unfortunately, the update- and query-time requirements of their scheme can easily become problematic for real-time monitoring applications, since the whole data structure must be "touched" for each update, and every wavelet coefficient queried to find the best few. Although [11] tries to reduce this cost by introducing more complex range-summable hash functions to make estimating individual wavelet coefficients faster, the number of queries does not decrease, and the additional complexity of the hash functions means that the update time increases further. Clearly, such high query times are not acceptable for any real-time monitoring environment, and pose the key obstacle in extending the algorithms in [11] to multi-dimensional data (where the domain size grows exponentially with data dimensionality).

In this paper, we propose the first known streaming algorithms for *space- and time-efficient tracking* of approximate wavelet summaries for both *one- and multi-dimensional data streams*. Our approach relies on a novel, sketch-based stream synopsis structure, termed the *Group-Count Sketch (GCS)* that allows us to provide similar space/accuracy tradeoffs as the simple sketches of [11], while guaranteeing: (1) small, logarithmic update times (essentially touching only a small fraction of the GCS for each streaming update) with simple, fast, hash functions; and, (2) polylogarithmic query times for computing the top wavelet coefficients from the GCS. In brief, our GCS algorithms rely on two key, novel technical ideas. First, we work *entirely in the wavelet domain*, in the sense that we directly sketch *wavelet coefficients*, rather than the original data vector, as updates arrive. Second, our GCSs employ *group structures based on hashing and hierarchical decomposition* over the wavelet domain to enable fast updates and efficient binary-search-like techniques for identifying the top wavelet coefficients in sublinear time. We also demonstrate that, by varying the degree of our search procedure, we can effectively explore the tradeoff between update and query costs in our GCS synopses. Our GCS algorithms and results also naturally extend to both the standard and non-standard form of the *multi-dimensional* wavelet transform, essentially providing the only known efficient solution for streaming wavelets in more than one dimension. As

[1] One issue surrounding using very small space is whether the data structure fits into the faster cache memory, which again emphasizes the importance of running time costs.

our experimental results with both synthetic and real-life data demonstrate, our GCS synopses allow very fast update and searching, capable of supporting very high speed data sources.

2 Preliminaries

In this section, we first discuss the basic elements of our stream-processing model and briefly introduce AMS sketches [2]; then, we present a short introduction to the Haar wavelet decomposition in both one and multiple dimensions, focusing on some of its key properties for our problem setting.

2.1 Stream Processing Model and Stream Sketches

Our input comprises a continuous stream of update operations, rendering a data vector a of N values (i.e., the data-domain size). Without loss of generality, we assume that the index of our data vector takes values in the integer domain $[N] = \{0, \ldots, N-1\}$, where N is a power of 2 (to simplify the notation). Each streaming update is a pair of the form $(i, \pm v)$, denoting a net change of $\pm v$ in the $a[i]$ entry; that is, the effect of the update is to set $a[i] \leftarrow a[i] \pm v$. Intuitively, "$+v$" ("$-v$") can be seen as v insertions (resp., deletions) of the i^{th} vector element, but more generally we allow entries to take negative values. (Our model instantiates the most general and, hence, most demanding *turnstile model* of streaming computations [20].) Our model generalizes to multi-dimensional data: for d data dimensions, a is a d-dimensional vector (*tensor*) and each update $((i_1, \ldots, i_d), \pm v)$ effects a net change of $\pm v$ on entry $a[i_1, \ldots, i_d]$.[2]

In the data-streaming context, updates are only seen *once in the (fixed) order of arrival*; furthermore, the rapid data-arrival rates and large data-domain size N make it impossible to store a explicitly. Instead, our algorithms can only maintain a concise *synopsis* of the stream that requires only sublinear space, and, at the same time, can (a) be maintained in small, sublinear processing time per update, and (b) provide query answers in sublinear time. Sublinear here means polylogarithmic in N, the data-vector size. (More strongly, our techniques guarantee update times that are sublinear in the *size of the synopsis*.)

Randomized AMS Sketch Synopses for Streams. The randomized *AMS sketch* [2] is a broadly applicable stream synopsis structure based on maintaining randomized linear projections of the streaming input data vector a. Briefly, an *atomic AMS sketch* of a is simply the *inner product* $\langle a, \xi \rangle = \sum_i a[i]\xi(i)$, where ξ denotes a random vector of four-wise independent ± 1-valued random variates. Such variates can be easily generated on-line through standard pseudo-random hash functions $\xi()$ using only $O(\log N)$ space (for seeding) [2, 11]. To maintain this inner product over the stream of updates to a, initialize a running counter X to 0 and set $X \leftarrow X \pm v\xi(i)$ whenever the update $(i, \pm v)$ is seen in the input stream. An *AMS sketch* of a comprises several independent

[2] Without loss of generality we assume a domain of $[N]^d$ for the d-dimensional case — different dimension sizes can be handled in a straightforward manner. Further, our methods do not need to know the domain size N beforehand — standard adaptive techniques can be used.

atomic AMS sketches (i.e., randomized counters), each with a different random hash function $\xi()$. The following theorem summarizes the key property of AMS sketches for stream-query estimation, where $||v||_2$ denotes the L_2-norm of a vector v, so $||v||_2 = \sqrt{\langle v, v\rangle} = \sqrt{\sum_i v[i]^2}$.

Theorem 1 ([1,2]). *Consider two (possibly streaming) data vectors a and b, and let Z denote the $O(\log(1/\delta))$-wise median of $O(1/\epsilon^2)$-wise means of independent copies of the atomic AMS sketch product $(\sum_i a[i]\xi_j(i))(\sum_i b[i]\xi_j(i))$. Then, $|Z - \langle a, b\rangle| \leq \epsilon||a||_2||b||_2$ with probability $\geq 1 - \delta$.*

Thus, using AMS sketches comprising only $O(\frac{\log(1/\delta)}{\epsilon^2})$ atomic counters we can approximate the vector inner product $\langle a, b\rangle$ to within $\pm\epsilon||a||_2||b||_2$ (hence implying an ϵ-relative error estimate for $||a||_2^2$).

2.2 Discrete Wavelet Transform Basics

The *Discrete Wavelet Transform (DWT)* is a useful mathematical tool for hierarchically decomposing functions in ways that are both efficient and theoretically sound. Broadly speaking, the wavelet decomposition of a function consists of a coarse overall approximation together with detail coefficients that influence the function at various scales [22]. *Haar wavelets* represent the simplest DWT basis: they are conceptually simple, easy to implement, and have proven their effectiveness as a data-summarization tool in a variety of settings [4, 24, 10].

One-Dimensional Haar Wavelets. Consider the one-dimensional data vector $a = [2, 2, 0, 2, 3, 5, 4, 4]$ ($N = 8$). The Haar DWT of a is computed as follows. We first average the values together pairwise to get a new "lower-resolution" representation of the data with the pairwise averages $[\frac{2+2}{2}, \frac{0+2}{2}, \frac{3+5}{2}, \frac{4+4}{2}] = [2, 1, 4, 4]$. This averaging loses some of the information in a. To restore the original a values, we need *detail coefficients*, that capture the missing information. In the Haar DWT, these detail coefficients are the differences of the (second of the) averaged values from the computed pairwise average. Thus, in our simple example, for the first pair of averaged values, the detail coefficient is 0 since $\frac{2-2}{2} = 0$, for the second it is -1 since $\frac{0-2}{2} = -1$. No information is lost in this process – one can reconstruct the eight values of the original data array from the lower-resolution array containing the four averages and the four detail coefficients. We recursively apply this pairwise averaging and differencing process on the lower-resolution array of averages until we reach the overall average, to get the full Haar decomposition. The final Haar DWT of a is given by $w_a = [11/4, -5/4, 1/2, 0, 0, -1, -1, 0]$, that is, the overall average followed by the detail coefficients in order of increasing resolution. Each entry in w_a is called a *wavelet coefficient*. The main advantage of using w_a instead of the original data vector a is that for vectors containing similar values most of the detail coefficients tend to have very small values. Thus, eliminating such small coefficients from the wavelet transform (i.e., treating them as zeros) introduces only small errors when reconstructing the original data, resulting in a very effective form of lossy data compression [22].

A useful conceptual tool for visualizing and understanding the (hierarchical) Haar DWT process is the *error tree* structure [19] (shown in Fig. 1(a) for our example

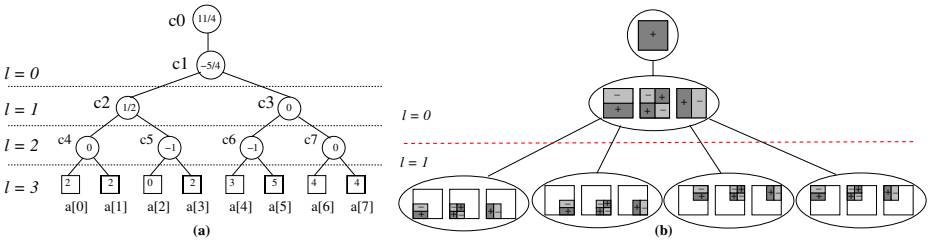


Fig. 1. Example error-tree structures for (a) a one-dimensional data array ($N = 8$), and (b) non-standard two-dimensional Haar coefficients for a 4×4 data array (coefficient magnitudes are multiplied by $+1$ (-1) in the "+" (resp., "-") labeled ranges, and 0 in blank areas)

array a). Each internal tree node c_i corresponds to a wavelet coefficient (with the root node c_0 being the overall average), and leaf nodes $a[i]$ correspond to the original data-array entries. This view allows us to see that the reconstruction of any $a[i]$ depends only on the $\log N + 1$ coefficients in the path between the root and $a[i]$; symmetrically, it means a change in $a[i]$ only impacts its $\log N + 1$ ancestors in an easily computable way. We define the *support* for a coefficient c_i as the contiguous range of data-array that c_i is used to reconstruct (i.e., the range of data/leaf nodes in the subtree rooted at c_i). Note that the supports of all coefficients at resolution level l of the Haar DWT are exactly the 2^l (disjoint) *dyadic ranges* of size $N/2^l = 2^{\log N - l}$ over $[N]$, defined as $R_{l,k} = [k \cdot 2^{\log N - l}, \ldots, (k+1) \cdot 2^{\log N - l} - 1]$ for $k = 0, \ldots, 2^l - 1$ (for each resolution level $l = 0, \ldots, \log N$). The Haar DWT can also be conceptualized in terms of vector inner-product computations: let $\phi_{l,k}$ denote the vector with $\phi_{l,k}[i] = 2^{l - \log N}$ for $i \in R_{l,k}$ and 0 otherwise, for $l = 0, \ldots, \log N$ and $k = 0, \ldots, 2^l - 1$; then, each of the coefficients in the Haar DWT of a can be expressed as the inner product of a with one of the N distinct Haar *wavelet basis vectors*:

$$\{\frac{1}{2}(\phi_{l+1,2k} - \phi_{l+1,2k+1}) : l = 0, \ldots, \log N - 1; k = 0, \ldots, 2^l - 1\} \cup \{\phi_{0,0}\}$$

Intuitively, wavelet coefficients with larger support carry a higher weight in the reconstruction of the original data values. To equalize the importance of all Haar DWT coefficients, a common normalization scheme is to scale the coefficient values at level l (or, equivalently, the basis vectors $\phi_{l,k}$) by a factor of $\sqrt{N/2^l}$. This normalization essentially turns the Haar DWT basis vectors into an *orthonormal basis* — letting c_i^* denote the normalized coefficient values, this fact has two important consequences: (1) The *energy* of the a vector is preserved in the wavelet domain, that is, $||a||_2^2 = \sum_i a[i]^2 = \sum_i (c_i^*)^2$ (by Parseval's theorem); and, (2) Retaining the B largest coefficients in terms of *absolute normalized value* gives the (provably) best B-term approximation in terms of Sum-Squared-Error (SSE) in the data reconstruction (for a given budget of coefficients B) [22].

Multi-Dimensional Haar Wavelets. There are two distinct ways to generalize the Haar DWT to the multi-dimensional case, the *standard* and *nonstandard* Haar decomposition [22]. Each method results from a natural generalization of the one-dimensional decomposition process described above, and both have been used in a wide variety of applications. Consider the case where a is a d-dimensional data array, comprising N^d

entries. As in the one-dimensional case, the Haar DWT of a results in a d-dimensional wavelet-coefficient array w_a with N^d coefficient entries. The non-standard Haar DWT works in $\log N$ phases where, in each phase, *one step* of pairwise averaging and differencing is performed across each of the d dimensions; the process is then repeated recursively (for the next phase) on the quadrant containing the averages across all dimensions. The standard Haar DWT works in d phases where, in each phase, a *complete* 1-dimensional DWT is performed for each one-dimensional row of array cells along dimension k, for all $k = 1, \ldots, d$. (full details and efficient decomposition algorithms are in [4, 24].) The supports of non-standard d-dimensional Haar coefficients are d-dimensional hyper-cubes (over dyadic ranges in $[N]^d$), since they combine 1-dimensional basis functions from the same resolution levels across all dimensions. The cross product of a standard d-dimensional coefficient (indexed by, say, $(i_1, \ldots, i_d)$) is, in general a d-dimensional hyper-rectangle, given by the cross-product of the 1-dimensional basis functions corresponding to coefficient indexes $i_1, \ldots, i_d$.

Error-tree structures can again be used to conceptualize the properties of both forms of d-dimensional Haar DWTs. In the non-standard case, the error tree is essentially a quadtree (with a fanout of 2^d), where all internal non-root nodes contain 2^{d-1} coefficients that have the same support region in the original data array but with different quadrant signs (and magnitudes) for their contribution. For standard d-dimensional Haar DWT, the error-tree structure is essentially a "cross-product" of d one-dimensional error trees with the support and signs of coefficient $(i_1, \ldots, i_d)$ determined by the product of the component one-dimensional basis vectors (for $i_1, \ldots, d$). Fig. 1(b) depicts a simple example error-tree structure for the non-standard Haar DWT of a 2-dimensional 4×4 data array. It follows that updating a single data entry in the d-dimensional data array a impacts the values of $(2^d - 1) \log N + 1 = O(2^d \log N)$ coefficients in the non-standard case, and $(\log N + 1)^d = O(\log^d N)$ coefficients in the standard case. Both multi-dimensional decompositions preserve the orthonormality, thus retaining the largest B coefficient values gives a provably SSE-optimal B-term approximation of a.

3 Problem Formulation and Overview of Approach

Our goal is to continuously track a compact B-coefficient wavelet synopsis under our general, high-speed update-stream model. We require our solution to satisfy all three key requirements for streaming algorithms outlined earlier in this paper, namely: (1) sublinear synopsis space, (2) sublinear per-item update time, and (3) sublinear query time, where sublinear means polylogarithmic in the domain size N. As in [11], our algorithms return only an *approximate* synopsis comprising (at most) B Haar coefficients that is provably near-optimal (in terms of the captured energy of the underlying vector) assuming that our vector satisfies the *"small-B property"* (i.e., most of its energy is concentrated in a small number of Haar DWT coefficients) — this assumption is typically satisfied for most real-life data distributions [11].

The streaming algorithm presented by Gilbert et al. [11] (termed "GKMS" in the remainder of the paper) focuses primarily on the one-dimensional case. The key idea is to maintain an AMS sketch for the streaming data vector a (as discussed in Sec. 2.1). To produce the approximate B-term representation, GKMS employs the constructed

sketch of a to estimate the inner product of a with all wavelet basis vectors, essentially performing an exhaustive search over the space of all wavelet coefficients to identify important ones. Although techniques based on range-summable random variables constructed using Reed-Muller codes were proposed to reduce or amortize the cost of this exhaustive search by allowing the sketches of basis vectors to be computed more quickly, the overall query time for discovering the top coefficients remains superlinear in N (i.e., at least $\Omega(\frac{1}{\epsilon^2}N \log N)$), violating our third requirement. For large data domains, say $N = 2^{32} \approx 4$ billion (such as the IP address domain considered in [11]), a query can take a very long time: over an hour, even if a million coefficient queries can be answered per second! This essentially renders a direct extension of the GKMS technique to multiple dimensions infeasible since it implies an exponential explosion in query cost (requiring at least $O(N^d)$ time to cycle through all coefficients in d dimensions). In addition, the update cost of the GKMS algorithm is *linear in the size of the sketch* since the whole data structure must be "touched" for each update. This is problematic for high-speed data streams and/or even moderate sized sketch synopses.

Our Approach. Our proposed solution relies on two key novel ideas to avoid the shortcomings of the GKMS technique. First, we work *entirely in the wavelet domain*: instead of sketching the original data entries, our algorithms sketch the wavelet-coefficient vector w_a as updates arrive. This avoids any need for complex range-summable hash functions. Second, we employ *hash-based grouping* in conjunction with *efficient binary-search-like techniques* to enable very fast updates as well as identification of important coefficients in polylogarithmic time.

– *Sketching in the Wavelet Domain.* Our first technical idea relies on the observation that we can efficiently produce sketch synopses of the stream *directly in the wavelet domain*. That is, we translate the impact of each streaming update on the relevant wavelet coefficients. By the linearity properties of the DWT and our earlier description, we know that an update to the data entries corresponds to only polylogarithmically many coefficients in the wavelet domain. Thus, on receiving an update to a, our algorithms directly convert it to $O(\text{polylog}(N))$ updates to the wavelet coefficients, and maintain an approximate representation of the wavelet coefficient vector w_a.

– *Time-Efficient Updates and Large-Coefficient Searches.* Sketching in the wavelet domain means that, at query time, we have an approximate representation of the wavelet-coefficient vector w_a and need to be able to identify all those coefficients that are "large", relative to the total energy of the data $\|w_a\|_2^2 = \|a\|_2^2$. While AMS sketches can give us these estimates (a point query is just a special case of an inner product), querying remains much too slow taking at least $\Omega(\frac{1}{\epsilon^2}N)$ time to find which of the N coefficients are the B largest. Note that although a lot of earlier work has given efficient streaming algorithms for identifying high-frequency items [5, 6, 18], our requirements here are quite different. Our techniques must monitor items (i.e., DWT coefficients) whose values increase and decrease over time, and which may very well be *negative* (even if all the data entries in a are positive). Existing work on "heavy-hitter" tracking focuses solely on non-negative frequency counts [6] often assumed to be non-decreasing over time [5, 18]. More strongly, we must find items whose *squared value* is a large

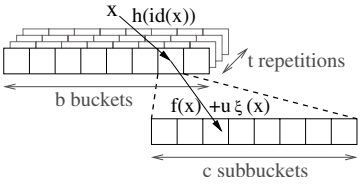


Fig. 2. Our Group-Count Sketch (GCS) data structure: x is hashed (t times) to a bucket and then to a subbucket within the bucket, where a counter is updated

fraction of the total vector energy $||w_a||_2^2$: this is a stronger condition since such "L_2^2 heavy hitters" may not be heavy hitters under the conventional sum-of-counts definition. [3]

At a high level, our algorithms rely on a *divide-and-conquer* or *binary-search-like* approach for finding the large coefficients. To implement this, we need the ability to efficiently estimate sums-of-squares for *groups* of coefficients, corresponding to dyadic subranges of the domain $[N]$. We then disregard low-energy regions and recurse only on high-energy groups — note that this guarantees no false negatives, as a group that contains a high-energy coefficient will also have high energy as a whole. Furthermore, our algorithms also employ *randomized, hash-based grouping* of dyadic groups and coefficients to guarantee that each update only touches a small portion of our synopsis, thus guaranteeing very fast update times.

4 Our Solution: The GCS Synopsis and Algorithms

We introduce a novel, hash-based probabilistic synopsis data structure, termed *Group-Count Sketch (GCS)*, that can estimate the energy (squared L_2 norm) of fixed groups of elements from a vector w of size N under our streaming model. (To simplify the exposition we initially focus on the one-dimensional case, and present the generalization to multiple dimensions later in this section.) Our GCS synopsis requires small, sublinear space and takes sublinear time to process each stream update item; more importantly, we can use a GCS to obtain a high-probability estimate of the energy of a group within additive error $\epsilon||w||_2^2$ in *sublinear time*. We then demonstrate how to use GCSs as the basis of efficient streaming procedures for tracking large wavelet coefficients.

Our approach takes inspiration from the AMS sketching solution for vector L_2-norm estimation; still, we need a much stronger result, namely the ability to estimate L_2 norms for a (potentially large) number of *groups of items* forming a partition of the data domain $[N]$. A simple solution would be to keep an AMS sketch of each group separately; however, there can be *many* groups, linear in N, and we cannot afford to devote this much space to the problem. We must also process streaming updates as quickly as possible. Our solution is to maintain a structure that first partitions items of w into their group, and then maps groups to buckets using a hash function. Within each bucket, we apply a second stage of hashing of items to sub-buckets, each containing an atomic AMS sketch counter, in order to estimate the L_2 norm of the bucket. In our

[3] For example, consider a set of items with counts $\{4, 1, 1, 1, 1, 1, 1, 1, 1\}$. The item with count 4 represents $\frac{2}{3}$ of the sum of the squared counts, but only $\frac{1}{3}$ of the sum of counts.

analysis, we show that this approach allows us to provide accurate estimates of the energy of any group in w with tight $\pm\epsilon||w||_2^2$ error guarantees.

The GCS Synopsis. Assume a total of k groups of elements of w that form a partition of $[N]$. For notational convenience, we use a function id that identifies the specific group that an element belongs to, $\texttt{id} : [N] \rightarrow [k]$. (In our setting, groups correspond to fixed dyadic ranges over $[N]$ so the id mapping is trivial.) Following common data-streaming practice, we first define a basic randomized estimator for the energy of a group, and prove that it returns a good estimate (i.e., within $\pm\epsilon||w||_2^2$ additive error) with constant probability $> \frac{1}{2}$; then, by taking the median estimate over t independent repetitions, we are able to reduce the probability of a bad estimate to exponentially small in t. Our basic estimator first hashes groups into b buckets and then, within each bucket, it hashes into c sub-buckets. (The values of t, b, and c parameters are determined in our analysis.) Furthermore, as in AMS sketching, each item has a $\{\pm 1\}$ random variable associated with it. Thus, our GCS synopsis requires three sets of t hash functions, $h_m : [k] \rightarrow [b]$, $f_m : [N] \rightarrow [c]$, and $\xi_m : [N] \rightarrow \{\pm 1\}$ ($m = 1, \ldots, t$). The randomization requirement is that h_m's and f_m's are drawn from families of pairwise independent functions, while ξ_m's are four-wise independent (as in basic AMS); such hash functions are easy to implement, and require only $O(\log N)$ bits to store.

Our GCS synopsis s consists of $t \cdot b \cdot c$ counters (i.e., atomic AMS sketches), labeled $s[1][1][1]$ through $s[t][b][c]$, that are maintained and queried as follows:

UPDATE(i, u). Set $s[m][h_m(\texttt{id}(i))][f_m(i)] += u \cdot \xi_m(i)$, for each $m = 1, \ldots, t$.

ESTIMATE(GROUP). Return the estimate $\text{median}_{m=1,\ldots,t} \sum_{j=1}^{c} (s[m][h_m(\text{GROUP})][j])^2$ for the energy of the group of items GROUP $\in \{1, \ldots, k\}$ (denoted by $\|\text{GROUP}\|_2^2$).

Thus, the update and query times for a GCS synopsis are simply $O(t)$ and $O(t \cdot c)$, respectively. The following theorem summarizes our key result for GCS synopses.

Theorem 2. *Our Group-Count Sketch algorithms estimate the energy of item groups of the vector w within additive error $\epsilon||w||_2^2$ with probability $\geq 1 - \delta$ using space of $O\left(\frac{1}{\epsilon^3} \log \frac{1}{\delta}\right)$ counters, per-item update time of $O\left(\log \frac{1}{\delta}\right)$, and query time of $O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$.*

Proof. Fix a particular group GROUP and a row r in the GCS; we drop the row index m in the context where it is understood. Let BUCKET be the set of elements that hash into the same bucket as GROUP does: $\text{BUCKET} = \{i \mid i \in [1, n] \wedge h(\texttt{id}(i)) = h(\text{GROUP})\}$. Among those, let COLL be the set of elements other than those of GROUP: $\text{COLL} = \{i \mid i \in [1, n] \wedge \texttt{id}(i) \neq \text{GROUP} \wedge h(\texttt{id}(i)) = h(\text{GROUP})\}$. In the following, we abuse notation in that we refer to a refer to both a group and the set of items in the group with the same name. Also, we write $\|S\|_2^2$ to denote the sum of squares of the elements (i.e. L_2^2) in set S: $\|S\|_2^2 = \sum_{i \in S} w[i]^2$.

Let est be the estimator for the sum of squares of the items of GROUP. That is, $est = \sum_{j=1}^{c} est_j$ where $est_j = (s[m][h_m(\text{GROUP})][j])^2$ is the square of the count in sub-bucket SUB_j. The expectation of this estimator is, by simple calculation, the sum of squares of items in sub-bucket j, which is a fraction of the sum of squares of the bucket. Similarly, using linearity of expectation and the four-wise independence of the ξ hash functions, the variance of est is bounded in terms of the square of the expectation:

$$\mathsf{E}[est] = \mathsf{E}[\|\text{BUCKET}\|_2^2] \qquad \mathsf{Var}[est] \leq \tfrac{2}{c}\mathsf{E}[\|\text{BUCKET}\|_2^4]$$

To calculate $\mathsf{E}[\|\text{BUCKET}\|_2^2]$, observe that the bucket contains items of GROUP as well as items from other groups denoted by the set COLL which is determined by h. Because of the pairwise independence of h, this expectation is bounded by a fraction of the total energy. Therefore:

$$\begin{aligned}\mathsf{E}[\|\text{BUCKET}\|_2^2] &= \|\text{GROUP}\|_2^2 + E[\|\text{COLL}\|_2^2] \leq \|\text{GROUP}\|_2^2 + \tfrac{1}{b}||w||_2^2 \\ \text{and } \mathsf{E}[\|\text{BUCKET}\|_2^4] &= \|\text{GROUP}\|_2^4 + \mathsf{E}[\|\text{COLL}\|_2^4] + 2\|\text{GROUP}\|_2^2\mathsf{E}[\|\text{COLL}\|_2^2] \\ &\leq ||w||_2^4 + \tfrac{1}{b}||w||_2^4 + 2||w||_2^2 \cdot \tfrac{1}{b}||w||_2^2 \leq (1+\tfrac{3}{b})||w||_2^4 \leq 2||w||_2^2\end{aligned}$$

since $\|\text{GROUP}\|_2^2 \leq ||w||_2^2$ and $b \geq 3$. The estimator's expectation and variance satisfy

$$\mathsf{E}[est] \leq \|\text{GROUP}\|_2^2 + \tfrac{1}{b}||w||_2^2 \qquad \mathsf{Var}[est] \leq \tfrac{4}{c}\|w\|_2^4$$

Applying the Chebyshev inequality we obtain $\mathsf{Pr}\left[|est - \mathsf{E}[est]| \geq \lambda||w||_2^2\right] \leq \dfrac{4}{c\lambda^2}$ and by setting $c = \frac{32}{\lambda^2}$ the bound becomes $\frac{1}{8}$, for some parameter λ. Using the above bounds on variance and expectation and the fact that $|x-y| \geq ||x|-|y||$ we have,

$$|est - \mathsf{E}[est]| \geq \left|est - \|\text{GROUP}\|_2^2 - \frac{1}{b}||w||_2^2\right| \geq \left||est - \|\text{GROUP}\|_2^2| - \frac{1}{b}||w||_2^2\right|.$$

Consequently (note that $\mathsf{Pr}[|x| > y] \geq \mathsf{Pr}[x > y]$),

$$\mathsf{Pr}\left[\left||est - \|\text{GROUP}\|_2^2| - \frac{1}{b}||w||_2^2 \geq \lambda||w||_2^2\right] \leq \mathsf{Pr}\left[|est - \mathsf{E}[est]| \geq \lambda||w||_2^2\right] \leq \frac{1}{8}$$

or equivalently, $\mathsf{Pr}\left[\left|est - \|\text{GROUP}\|_2^2\right| \geq \left(\lambda + \frac{1}{b}\right)||w||_2^2\right] \leq \frac{1}{8}$. Setting $b = \frac{1}{\lambda}$ we get $\mathsf{Pr}\left[\left|est - \|\text{GROUP}\|_2^2\right| \geq 2\lambda||w||_2^2\right] \leq \frac{1}{8}$ and to obtain an estimator with $\epsilon||w||_2^2$ additive error we require $\lambda = \frac{\epsilon}{2}$ which translates to $b = O(\frac{1}{\epsilon})$ and $c = O(\frac{1}{\epsilon^2})$.

By Chernoff bounds, the probability that the median of t independent instances of the estimator deviates by more than $\epsilon||w||_2^2$ is less than e^{-qt}, for some constant q. Setting this to the probability of failure δ, we require $t = O\left(\log \frac{1}{\delta}\right)$, which gives the claimed bounds. □

Hierarchical Search Structure for Large Coefficients. We apply our GCS synopsis and estimators to the problem of finding items with large energy (i.e., squared value) in the w vector. Since our GCS works in the wavelet domain (i.e., sketches the wavelet coefficient vector), this is exactly the problem of recovering important coefficients. To efficiently recover large-energy items, we impose a regular tree structure on top of the data domain $[N]$, such that every node has the same degree r. Each level in the tree induces a partition of the nodes into groups corresponding to *r-adic ranges*, defined by the nodes at that level. [4] For instance, a binary tree creates groups corresponding to dyadic ranges of size 1, 2, 4, 8, and so on. The basic idea is to perform a search over the tree for those high-energy items above a specified energy threshold, $\phi||w||_2^2$. Following the discussion in Section 3, we can prune groups with energy below the threshold and, thus, avoid looking inside those groups: if the estimated energy is accurate, then these cannot contain any high-energy elements. Our key result is that, using such a hierarchical search structure of GCSs, we can provably (within appropriate probability bounds) retrieve all items above the threshold plus a controllable error quantity ($(\phi+\epsilon)||w||_2^2$), and retrieve no elements below the threshold minus that small error quantity ($(\phi-\epsilon)||w||_2^2$).

[4] Thus, the `id` function for level l is easily defined as $\texttt{id}_l(i) = \lfloor i/r^l \rfloor$.

Theorem 3. *Given a vector w of size N we can report, with high probability $\geq 1-\delta$, all elements with energy above $(\phi+\epsilon)||w||_2^2$ (where $\phi \geq \epsilon$) within additive error of $\epsilon||w||_2^2$ (and therefore, report no item with energy below $(\phi-\epsilon)||w||_2^2$) using space of $O\left(\frac{\log_r N}{\epsilon^3}\cdot\log\frac{r\log_r N}{\phi\delta}\right)$, per item processing time of $O\left(\log_r N\cdot\log\frac{r\log_r N}{\phi\delta}\right)$ and query time of $O\left(\frac{r}{\phi\epsilon^2}\cdot\log_r N\cdot\log\frac{r\log_r N}{\phi\delta}\right)$.*

Proof. Construct $\log_r N$ GCSs (with parameters to be determined), one for each level of our r-ary search-tree structure. We refer to an element that has energy above $\phi||w||_2^2$ as a "hot element", and similarly groups that have energy above $\phi||w||_2^2$ as "hot ranges". The key observation is that all r-adic ranges that contain a hot element are also hot. Therefore, at each level (starting with the root level), we identify hot r-adic ranges by examining only those r-adic ranges that are contained in hot ranges of the previous level. Since there can be at most $\frac{1}{\phi}$ hot elements, we only have to examine at most $\frac{1}{\phi}\log_r N$ ranges and pose that many queries. Thus, we require the failure probability to be $\frac{\log_r N}{\phi\delta}$ for each query so that, by the union bound, we obtain a failure probability of at most δ for reporting all hot elements. Further, we require each level to be accurate within $\epsilon||w||_2^2$ so that we obtain all hot elements above $(\phi+\epsilon)||w||_2^2$ and none below $(\phi-\epsilon)||w||_2^2$. The theorem follows. □

Setting the value of r gives a tradeoff between query time and update time. Asymptotically, we see that the update time decreases as the degree of the tree structure, r, increases. This becomes more pronounced in practice, since it usually suffices to set t, the number of tests, to a small constant. Under this simplification, the update cost essentially reduces to $O(\log_r N)$, and the query time reduces to $O(\frac{r}{\epsilon^2\phi}\log_r N)$. (We will see this clearly in our experimental analysis.) The extreme settings of r are 2 and N: $r=2$ imposes a binary tree over the domain, and gives the fastest query time but $O(\log_2 N)$ time per update; $r=N$ means updates are effectively constant $O(1)$ time, but querying requires probing the whole domain, a total of N tests to the sketch.

Sketching in the Wavelet Domain. As discussed earlier, given an input update stream for data entries in a, our algorithms build GCS synopses on the corresponding wavelet coefficient vector w_a, and then employ these GCSs to quickly recover a (provably good) approximate B-term wavelet representation of a. To accomplish the first step, we need an efficient way of "translating" updates in the original data domain to the domain of wavelet coefficients (for both one- and multi-dimensional data streams).

– *One-Dimensional Updates.* An update (i,v) on a translates to the following collection of $\log N+1$ updates to wavelet coefficients (that lie on the path to leaf $a[i]$, Fig. 1(a)):
$\left(0, 2^{-\frac{1}{2}\log N}v\right), \left\{\left(2^{\log N-l}+k, (-1)^{k \mod 2}2^{-\frac{l}{2}}v\right)\text{: for each } l=0,\ldots,\log N-1\right\}$,
where $l=0,\ldots,\log N-1$ indexes the resolution level, and $k=\lfloor i2^{-l}\rfloor$. Note that each coefficient update in the above set is easily computed in constant time.

– *Multi-Dimensional Updates.* We can use exactly the same reasoning as above to produce a collection of (constant-time) wavelet-coefficient updates for a given data update in d dimensions (see, Fig. 1(b)). As explained in Section 2.2, the size of this collection of updates in the wavelet domain is $O(\log^d N)$ and $O(2^d\log N)$ for standard and

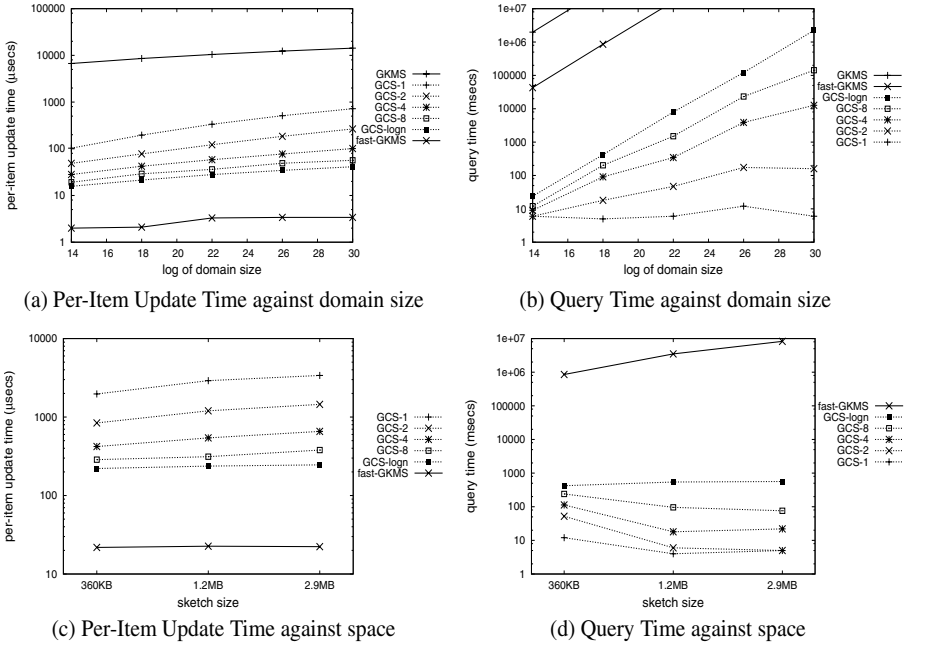
non-standard Haar wavelets, respectively. A subtle issue here is that our search-tree structure operates over a linear ordering of the N^d coefficients, so we require a fast method for linearizing the multi-dimensional coefficient array — any simple linearization technique will work (e.g., row-major ordering or other space-filling curves).

Using GCSs for Approximate Wavelets. Recall that our goal is to (approximately) recover the B most significant Haar DWT coefficients, without exhaustively searching through all coefficients. As shown in Theorem 3, creating GCSs for for dyadic ranges over the (linearized) wavelet-coefficient domain, allows us to efficiently identify high-energy coefficients. (For simplicity, we fix the degree of our search structure to $r = 2$ in what follows.) An important technicality here is to select the right threshold for coefficient energy in our search process, so that our final collection of recovered coefficients provably capture most of the energy in the optimal B-term representation. Our analysis in the following theorem shows how to set this threshold, an proves that, for data vectors satisfying the "small-B property", our GCS techniques can efficiently track near-optimal approximate wavelet representations. (We present the result for the standard form of the multi-dimensional Haar DWT — the one-dimensional case follows as the special case $d = 1$.)

Theorem 4. *If a d-dimensional data stream over the $[N]^d$ domain has a B-term standard wavelet representation with energy at least $\eta||a||_2^2$, where $||a||_2^2$ is the entire energy, then our GCS algorithms can estimate an at-most-B-term standard wavelet representation with energy at least $(1-\epsilon)\eta||a||_2^2$ using space of $O(\frac{B^3 d \log N}{\epsilon^3 \eta^3} \cdot \log \frac{Bd \log N}{\epsilon\eta\delta})$, per item processing time of $O(d \log^{d+1} N \cdot \log \frac{Bd \log N}{\epsilon\eta\delta})$, and query time of $O(\frac{B^3 d}{\epsilon^3\eta^3} \cdot \log N \cdot \log \frac{Bd \log N}{\epsilon\eta\delta})$.*

Proof. Use our GCS search algorithm and Theorem 3 to find all coefficients with energy at least $\frac{\epsilon\eta}{B}||a||_2^2 = \frac{\epsilon\eta}{B}||w||_2^2$. (Note that $||a||_2^2$ can be easily estimated to within small relative error from our GCSs.) Among those choose the highest B coefficients; note that there could be less than B found. For those coefficients selected, observe we incur two types of error. Suppose we choose a coefficient which is included in the best B-term representation, then we could be inaccurate by at most $\frac{\epsilon\eta}{B}||a||_2^2$. Now, suppose we choose coefficient c_1 which is not in the best B-term representation. There has to be a coefficient c_2 which is in the best B-term representation, but was rejected in favor of c_1. For this rejection to have taken place their energy must differ by at most $2\frac{\epsilon\eta}{B}||a||_2^2$ by our bounds on the accuracy of estimation for groups of size 1. Finally, note that for any coefficient not chosen (for the case when we pick fewer than B coefficients) its true energy must be less than $2\frac{\epsilon\eta}{B}||a||_2^2$. It follows that the total energy we obtain is at most $2\epsilon\eta||a||_2^2$ less than that of the best B-term representation. Setting parameters λ, ϵ', N' of Theorem 3 to $\lambda = \epsilon' = \frac{\epsilon\eta}{B}$ and $N' = N^d$ we obtain the stated space and query time bounds. For the per-item update time, recall that a single update in the original data domain requires $O(\log^d N)$ coefficient updates. □

The corresponding result for the non-standard Haar DWT follows along the same lines. The only difference with Theorem 4 comes in the per-update processing time which, in the non-standard case, is $O(d2^d \log N \cdot \log \frac{Bd \log N}{\epsilon\eta\delta})$.



(a) Per-Item Update Time against domain size

(b) Query Time against domain size

(c) Per-Item Update Time against space

(d) Query Time against space

Fig. 3. Performance on one-dimensional data

5 Experiments

Data Sets and Methodology. We implemented our algorithms in a mixture of C and C++, for the Group-Count sketch (GCS) with variable degree. For comparison we also implemented the method of [11] (GKMS) as well as a modified version of the algorithm with faster update performance using ideas similar to those in the Group-Count sketch, which we denote by fast-GKMS. Experiments were performed on a 2GHz processor machine, with 1GB of memory. We worked with a mixture of real and synthetic data:

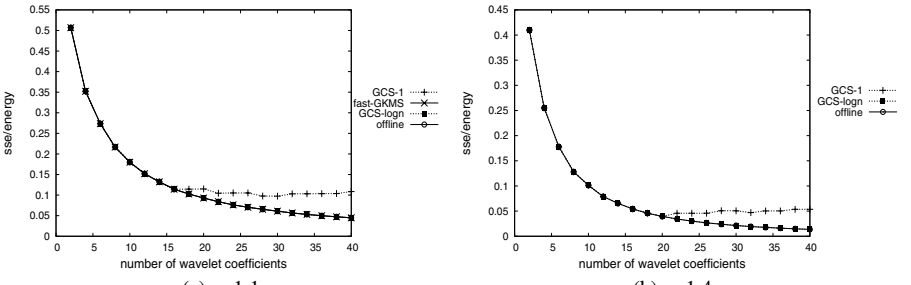
- *Synthetic Zipfian Data* was used to generate data from arbitrary domain sizes and with varying skewness. By default the skewness parameter of the distribution is $z = 1.1$.
- *Meteorological data set*[5] comprised of 10^5 meteorological measurements. These were quantized and projected appropriately to generate data sets with dimensionalities between 1 and 4. For the experiments described here, we primarily made use of the `AirTemperature` and `WindSpeed` attributes to obtain 1- and 2-dimensional data streams.

In our experiments, we varied the domain size, the size of the sketch[6] and the degree of the search tree of our GCS method and measured (1) per-item update time, (2) query

[5] `http://www-k12.atmos.washington.edu/k12/grayskies/`

[6] In each experiment, all methods are given the same total space to use.

time and (3) accuracy. In all figures, GCS-k denotes that the degree of the search tree is 2^k; i.e. GCS-1 uses a binary search tree, whereas GCS-logn uses an n-degree tree, and so has a single level consisting of the entire wavelet domain.

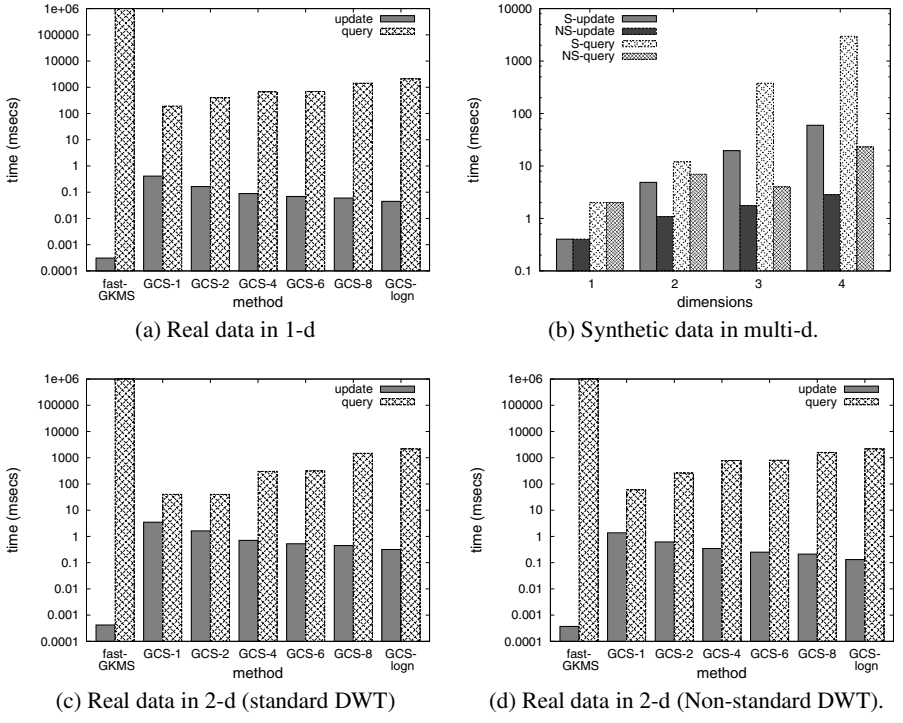


(a) z=1.1 (b) z=1.4

Fig. 4. Accuracy of Wavelet Synopses

One-Dimensional Experiments. In the first experimental setup we used a synthetic 1-dimensional data stream with updates following the Zipfian distribution ($z = 1.1$). Space was increased based on the log of the dimension, so for $\log N = 14$, 280KB was used, up to 600KB for $\log N = 30$. Figure 3 (a) shows the per-item update time for various domain sizes, and Figure 3 (b) shows the time required to perform a query, asking for the top-5 coefficients. The GKMS method takes orders of magnitude longer for both updates and queries, and this behavior is seen in all other experiments, so we do not consider it further. Apart from this, the ordering (fastest to slowest) is reversed between update time and query time. Varying the degree of the search tree allows update time and query time to be traded off. While the fast-GKMS approach is the fastest for updates, it is dramatically more expensive for queries, by several orders of magnitude. For domains of size 2^{22}, it takes several hours to recover the coefficients, and extrapolating to a 32 bit domain means recovery would take over a week. Clearly this is not practical for realistic monitoring scenarios. Although GCS-logn also performs exhaustive search over the domain size, its query times are significantly lower as it does not require a sketch construction and inner-product query per wavelet coefficient.

Figures 3 (c) and (d) show the performance as the sketch size is increased. The domain size was fixed to 2^{18} so that the fast-GKMS method would complete a query in reasonable time. Update times do not vary significantly with increasing space, in line with our analysis (some increase in cost may be seen due to cache effects). We also tested the accuracy of the approximate wavelet synopsis for each method. We measured the SSE-to-energy ratio of the estimated B-term synopses for varying B and varying zipf parameter and compared it against the optimal B-term synopsis computed offline. The results are shown in Figures 4 (a) and (b), where each sketch was given space 360KB. In accordance to analysis (GCS requires $O(\frac{1}{\epsilon})$ times more space to provide the same guarantees with GKMS) the GCS method is slightly less accurate when estimating more than the top-15 coefficients. However, experiments showed that increasing the size to 1.2MB resulted in equal accuracy. Finally we tested the performance of our methods



(a) Real data in 1-d

(b) Synthetic data in multi-d.

(c) Real data in 2-d (standard DWT)

(d) Real data in 2-d (Non-standard DWT).

Fig. 5. Performance on 1-d Real Data and multi-d Real and Synthetic Data

on single dimensional meteorological data of domain size 2^{20}. Per-item and query times in Figure 5 (a) are similar to those on synthetic data.

Multi-Dimensional Experiments. We compared the methods for both wavelet decomposition types in multiple dimensions. First we tested our GCS method for a synthetic dataset ($z = 1.1$, 10^5 tuples) of varying dimensionality. In Figure 5 (b) we kept the total domain size constant at 2^{24} while varying the dimensions between 1 and 4. The per-item update time is higher for the standard decomposition, as there are more updates on the wavelet domain per update on the original domain. The increase in query time can be attributed to the increasing sparseness of the domain as the dimensionality increases which makes searching for big coefficients harder. This is a well known effect of multidimensional standard and non-standard decompositions. For the real dataset, we focus on the two dimensional case; higher dimensions are similar. Figure 5(c) and (d) show results for the standard and non-standard respectively. The difference between GCS methods and fast-GKMS is more pronounced, because of the additional work in producing multidimensional wavelet coefficients, but the query times remain significantly less (query times were in the order of hours for fast-GKMS), and the difference becomes many times greater as the size of the data domain increases.

Experimental Summary. The Group-Count sketch approach is the only method that achieves reasonable query times to return an approximate wavelet representation of

data drawn from a moderately large domain (2^{20} or larger). Our first implementation is capable of processing tens to hundreds of thousands of updates per second, and giving the answer to queries in the order of a few seconds. Varying the degree of the search tree allows a tradeoff between query time and update time to be established. The observed accuracy is almost indistinguishable from the exact solution, and the methods extend smoothly to multiple dimensions with little degradation of performance.

6 Related Work

Wavelets have a long history of successes in the signal and image processing arena [16, 22] and, recently, they have also found their way into data-management applications. Matias et al. [19] first proposed the use of Haar-wavelet coefficients as synopses for accurately estimating the selectivities of range queries. Vitter and Wang [24] describe I/O-efficient algorithms for building multi-dimensional Haar wavelets from large relational data sets and show that a small set of wavelet coefficients can efficiently provide accurate approximate answers to range aggregates over OLAP cubes. Chakrabarti et al. [4] demonstrate the effectiveness of Haar wavelets as a general-purpose approximate query processing tool by designing efficient algorithms that can process complex relational queries (with joins, selections, etc.) entirely in the wavelet-coefficient domain. Schmidt and Shahabi [21] present techniques using the Daubechies family of wavelets to answer general polynomial range-aggregate queries. Deligiannakis and Roussopoulos [8] introduce algorithms for building wavelet synopses over data with multiple measures. Finally, I/O efficiency issues are studied by Jahangiri et al. [15] for both forms of the multi-dimensional DWT.

Interest in data streams has also increased rapidly over the last years, as more algorithms are presented that provide solutions in a streaming one-pass, low memory environment. Overviews of data-streaming issues and algorithms can be found, for instance, in [3, 20]. Sketches first appeared for estimating the second frequency moment of a set of elements [2] and have since proven to be a useful summary structure in such a dynamic setting. Their application includes uses for estimating join sizes of queries over streams [1, 9], maintaining wavelet synopses [11], constructing histograms [12, 23], estimating frequent items [5, 6] and quantiles [13]. The work of Gilbert et al. [11] for estimating the most significant wavelet coefficients is closely related to ours. As we discuss, the limitation is the high query time required for returning the approximate representation. In follow-up work, the authors proposed a more theoretical approach with somewhat improved worst case query times [12]. This work considers an approach based on a complex construction of range-summable random variables to build sketches from which wavelet coefficients can be obtained. The update times remain large. Our bounds improve those that follow from [12], and our algorithm is much simpler to implement. In similar spirit, Thaper et al. [23] use AMS sketches to construct an optimal B-bucket histogram of large multi-dimensional data. No efficient search techniques are used apart from an exhaustive greedy heuristic which always chooses the next best bucket to include in the histogram; still, this requires an exhaustive search over a huge space. The idea of using *group-testing* techniques to more efficiently find heavy items appears in several prior works [6, 7, 12]; here, we show that it is possible to apply similar

ideas to groups under L_2 norm, which has not been explored previously. Recently, different techniques have been proposed for constructing wavelet synopses that minimize non-Euclidean error metrics, under the time-series model of streams [14, 17].

7 Conclusions

We have proposed the first known streaming algorithms for space- and time-efficient tracking of approximate wavelet summaries for both one- and multi-dimensional data streams. Our approach relies on a novel, Group-Count Sketch (GCS) synopsis that, unlike earlier work, satisfies all three key requirements of effective streaming algorithms, namely: (1) polylogarithmic space usage, (2) small, logarithmic update times (essentially touching only a small fraction of the GCS for each streaming update); and, (3) polylogarithmic query times for computing the top wavelet coefficients from the GCS. Our experimental results with both synthetic and real-life data have verified the effectiveness of our approach, demonstrating the ability of GCSs to support very high speed data sources. As part of our future work, we plan to extend our approach to the problem of extended wavelets [8] and histograms [23].

References

1. N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. "Tracking join and self-join sizes in limited storage". In *ACM PODS*, 1999.
2. N. Alon, Y. Matias, and M. Szegedy. "The space complexity of approximating the frequency moments". In *ACM STOC*, 1996.
3. B. Babcock, S. Babu, M. Datar, R. Motwani, and Jennifer Widom. "Models and issues in data stream systems". In *ACM PODS*, 2002.
4. K. Chakrabarti, M. N. Garofalakis, R. Rastogi, and K. Shim. "Approximate query processing using wavelets". In *VLDB*, 2000.
5. M. Charikar, K. Chen, and M. Farach-Colton. "Finding frequent items in data streams". In *ICALP*, 2002.
6. G. Cormode and S. Muthukrishnan. "What's hot and what's not: Tracking most frequent items dynamically". In *ACM PODS*, 2003.
7. G. Cormode and S. Muthukrishnan. "What's new: Finding significant differences in network data streams". In *IEEE Infocom*, 2004.
8. A. Deligiannakis and N. Roussopoulos. "Extended wavelets for multiple measures". In *ACM SIGMOD*, 2003.
9. A. Dobra, M. N. Garofalakis, J. Gehrke, and R. Rastogi. "Processing complex aggregate queries over data streams". In *ACM SIGMOD*, 2002.
10. M. Garofalakis and A. Kumar. "Deterministic Wavelet Thresholding for Maximum-Error Metrics". In *ACM PODS*, 2004.
11. A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. "One-pass wavelet decomposition of data streams". *IEEE TKDE*, 15(3), 2003.
12. A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. "Fast, small-space algorithms for approximate histogram maintenance". In *ACM STOC*, 2002.
13. A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. "How to summarize the universe: Dynamic maintenance of quantiles". In *VLDB*, 2002.
14. S. Guha and B. Harb. "Wavelet Synopsis for Data Streams: Minimizing non-Euclidean Error" In *KDD*, 2005.

15. M. Jahangiri, D. Sacharidis, and C. Shahabi. "Shift-Split: I/O efficient maintenance of wavelet-transformed multidimensional data". In *ACM SIGMOD*, 2005.
16. B. Jawerth and W. Sweldens. "An Overview of Wavelet Based Multiresolution Analyses". *SIAM Review*, 36(3), 1994.
17. P. Karras and N. Mamoulis. "One-pass wavelet synopses for maximum-error metrics". In *VLDB*, 2005.
18. G.S. Manku and R. Motwani. "Approximate frequency counts over data streams". In *VLDB*, 2002.
19. Y. Matias, J.S. Vitter, and M. Wang. "Wavelet-based histograms for selectivity estimation". In *ACM SIGMOD*, 1998.
20. S. Muthukrishnan. Data streams: algorithms and applications. In *SODA*, 2003.
21. R.R. Schmidt and C. Shahabi. "Propolyne: A fast wavelet-based technique for progressive evaluation of polynomial range-sum queries". In *EDBT*, 2002.
22. E. J. Stollnitz, T. D. Derose, and D. H. Salesin. *"Wavelets for computer graphics: theory and applications"*. Morgan Kaufmann Publishers, 1996.
23. N. Thaper, S. Guha, P. Indyk, and N. Koudas. "Dynamic multidimensional histograms". In *ACM SIGMOD*, 2002.
24. J.S. Vitter and M. Wang. "Approximate computation of multidimensional aggregates of sparse data using wavelets". In *ACM SIGMOD*, 1999.