Improved implementation and experimental evaluation of the max-error optimized wavelet synopses

Yossi Matias
School of Computer Science
Tel Aviv University
matias@post.tau.ac.il

Daniel Urieli
School of Computer Science
Tel Aviv University
daniel1@post.tau.ac.il

(2004, rev: Jul 2005)

Abstract

This paper provides an improved implementation of an algorithm for building wavelet synopses for max-error metrics, recently introduced by Garofalakis and Kumar (GK) [4]. Given a storage space of size M, the GK algorithm finds a wavelet synopsis of size M, which minimizes the max (absolute or relative) error, measured over the data values, with respect to any other wavelet synopsis of size M. The running time of the GK algorithm is $O\left(N^2 M \log M\right)$ and its space complexity is $O\left(N^2 M\right)$. In this paper we improve the time and space complexities by a factor of M, reducing the running time to $O\left(N^2 \log M\right)$ and the space requirement to $O\left(N^2\right)$. As in [4] no experimental results were shown, we present experimental comparison between the accuracy of the GK synopsis with other wavelet synopses, as well as experimental comparison between the running-time of the original GK algorithm with our improved implementation. We also apply the GK synopsis for range-queries, built on the raw data as well as over the prefix-sums of the data, and compare it experimentally with other wavelet synopses, demonstrating an interesting similarity to another synopsis that can be computed in linear time.

1 Introduction

In recent years there has been increasing attention to the development and study of data synopses, as effective means for addressing performance issues in massive data sets. Data synopses are concise representations of data sets, that are meant to effectively support approximate queries to the represented data sets [5]. A primary constraint of a data synopsis is its size. The effectiveness of a data synopsis is measured by the accuracy of the answers it provides, as well as by its response time and its construction time. Several different synopses were introduced and studied, including random samples, sketches, and different types of histograms. Recently, wavelet-based synopses were introduced and shown to be a powerful tool for building effective data synopses for various applications, including selectivity estimation for

query optimization in DBMS, approximate query processing in OLAP applications and more (see [11, 17, 15, 16, 1, 3, 2, 4], and references therein).

The general idea of wavelet-based approximations is to transform a given data vector of size N into a representation with respect to a wavelet basis (this is called a *wavelet transform*), and approximate it using only $M \ll N$ wavelet basis vectors, by retaining only M coefficients from the linear combination that spans the data vector (*coefficients thresholding*). The linear combination that uses only M coefficients (and assumes that all other coefficients are zero) defines a new vector that approximates the original vector, using less space. This is called *M-term approximation*, which defines a *wavelet synopsis* of size M.

An M-term approximation, like any other approximation, creates a vector of approximation errors, which are the differences between the approximated data values and the original data values. Several works have dealt with providing wavelet-based synopses that are effective with respect to the mean-squared norm of error approximation. Recently, Garofalakis and Kumar [4] addressed the problem of minimizing the error approximation in the max-norm. They present a dynamic programming algorithm and find an optimal M-term approximation, which minimizes the max absolute or relative error over the data values. The time and space complexities of their algorithm are $O(N^2 M \log M)$ and $O(N^2 M)$, respectively, where N is the data size and M is the approximation size (the number of wavelet coefficients). Throughout the paper we call this algorithm the "GK algorithm" and the synopsis it builds the "GK synopsis".

1.1 Contributions

In this paper we provide an improved implementation of the GK algorithm. We also provide experimental evaluation of both the original and improved implementations of the GK algorithm, as well as of the accuracy of the GK synopsis. We improve the implementation of the algorithm and reduce its time and space complexities each by a factor of M, obtaining running time of $O(N^2 \log M)$ and space complexity of $O(N^2)$. The improvement is achieved using a rather simple observation. The original algorithm computed some redundant computations, which are executed due to the recursive nature of the GK algorithm. We avoid these computations, observing that most of the computed points are not required for the recursive computation.

We have conducted the following experimental evaluation:

- **Accuracy for point queries.** The GK synopsis minimizes the maximum relative or absolute error measured over the data values, or equivalently over all possible point queries. We compared it with the standard wavelet synopsis [11, 17]. When using the relative-error version of the GK synopsis we establish that its maximum-relative error can be 70 times smaller than the standard wavelet synopsis, which is not designed for max relative-error minimization. When comparing the absolute-error version of the GK synopsis, results were better than the standard synopsis, but with a smaller factor of 1.25–2.

- **Accuracy for range sum queries.** The paper [11, 17] introduced two methods for answering of range-sum queries using a wavelet synopsis. One method is answering range queries using wavelet transform that is done over the raw data. The other method is answering range queries using wavelet transform that is done over the prefix-sums of the original data. We implemented both methods for the GK synopsis and tested it

experimentally. We compared the relative-error version to a workload-based adaptive-greedy algorithm of [7, 13], which is adapted to minimize the mean-squared relative error (MRE) for workloads of range queries. We got that the adaptive greedy algorithm showed better results with respect to both MRE and the maximum-relative error measure over the tested data sets.

We compared the GK synopsis with the standard synopsis, both built over the raw data, and the standard synopsis was superior both with respect to the MSE and max-absolute error measures.

We also compared the GK synopsis, built over the prefix sums of the data, with a Range-Optimal Prefix-Sums (ROPS) synopsis for range-queries which we recently introduced in [9]. Despite of the fact that the ROPS synopsis and the GK synopsis are aimed at minimizing different error measures, and constructed differently, we got very similar results for both synopses, on the tested data sets, with respect to both MSE and max-absolute error measures.

- **Running time evaluation.** We compared the running time of both the original and improved implementation of the GK algorithm, showing that the running time of the improved implementation is faster by approximately a factor of M (the synopses size), as expected.

1.2 Related results

Recently, Muthukrishnan [12] improved the GK algorithm and reduced its running time to $O(N^2M/\log M)$, while leaving the space complexity unchanged. His work focuses on a related problem – minimizing the weighted L_2 norm of the vector of absolute errors, measured over all possible point queries, using the Haar wavelet basis [1] – showing that the GK algorithm can be adapted for this problem as well. Here we improve the complexity of the GK algorithm, reducing *both* its space and time, where the new time and space complexities are better than in [4, 12]. Our improved implementation can be adapted to address the workload based wavelet synopses problem for Haar wavelets, with the same complexities.

Recently, Guha [6] has obtained a related improvement. With a more careful acounting for the running time he has further reduced the time complexity by an additional factor of $\log B$.

2 Wavelet synopses based on the Haar basis

Haar wavelets are conceptually the simplest wavelet basis functions, and they are used in the original paper. To illustrate how Haar wavelets work, we will start with a simple example borrowed from [11, 17].

Suppose we have one-dimensional "signal" of $N = 8$ data items: $S = [2, 2, 0, 2, 3, 5, 4, 4]$. We will show how the Haar wavelet transform is done over S. We first average the signal values, pairwise, to get a new lower-resolution signal with values $[2, 1, 4, 4]$. That is, the first two values in the original signal (2 and 2) average to 2, and the second two values 0 and 2 average to 1,

[1] This problem is also known as the workload-based wavelet synopses problem, introduced by [7, 13]. We have recently provided a linear time solution using weighted Haar bases [10].

and so on. We also store the pairwise differences of the original values (divided by 2) as detail coefficients. In the above example, the four detail coefficients are $(2-2)/2 = 0$, $(0-2)/2 = -1$, $(3-5)/2 = -1$, and $(4-4)/2 = 0$. It is easy to see that the original values can be recovered from the averages and differences.

This was one phase of the Haar wavelet transform. By repeating this process recursively on the averages, we get the Haar wavelet transform (Table 1). We define the *wavelet transform* (also called *wavelet decomposition*) of the original eighth-value signal to be the single coefficient representing the overall average of the original signal, followed by the detail coefficients in the order of increasing resolution. Thus, for the one-dimensional Haar basis, the wavelet transform of our signal is given by
$\tilde{S} = [2\frac{3}{4}, -1\frac{1}{4}, \frac{1}{2}, 0, 0, -1, -1, 0]$

Resolution	Averages	Detail Coefficients
8	[2, 2, 0, 2, 3, 5, 4, 4]	
4	[2, 1, 4, 4]	[0,-1,-1, 0]
2	[1.5, 4]	[0.5, 0]
1	[2.75]	-1.25

Table 1: Haar Wavelet Decomposition

The individual entries are called the wavelet coefficients. The wavelet decomposition is very efficient computationally, requiring only *O(N)* CPU time and *O(N/B)* I/Os to compute for a signal of N values, where B is the disk-block size.
No information has been gained or lost by this process. The original signal has eight values, and so does the transform. Given the transform, we can reconstruct the exact signal by recursively adding and subtracting the detail coefficients from the next-lower resolution. In fact we have transformed the signal S into a representation with respect to another basis of R^8: The Haar wavelet basis. A detailed discussion can be found, for example, in [14].

A helpful tool for exploring and understanding the Haar wavelet transform is the error tree structure [11, 17], depicted in Figure 1. Understanding the error tree is crucial to understanding the algorithm of [4], and our improvement. The figure demonstrates how an original data value can be reconstructed in $O(\log N)$ time. The wavelet coefficients can be treated as nodes in the tree, where the original data are the leaves of the tree. In order to reconstruct a data value we just go down the tree, adding or subtracting coefficient values on the path to the leaf. When we go right we subtract the value (which is a pairwise difference divided by two, as described above), and when we go left we add the coefficient's value.

The next step is to keep only M of the N wavelet coefficients as our synopsis, assuming all other coefficients are zero, using the M coefficients to approximate the original data. When building an approximated vector using the synopsis, we can measure the approximation error in several ways. Let d_i be a data value, and $\hat{d}_i$ be the approximation of d_i, constructed by the synopsis. The *absolute error* over the data value is defined as $|d_i - \hat{d}_i|$, and the relative error is defined as $\frac{|d_i - \hat{d}_i|}{Max(d_i, s)}$, where s is a sanity bound preventing from small data values from dominating the relative error (for example, s can be taken to be 1). The approximation defines a vector of errors over all data values (equivalently point queries). Usually the purpose is to minimize some L_p norm of this vector.

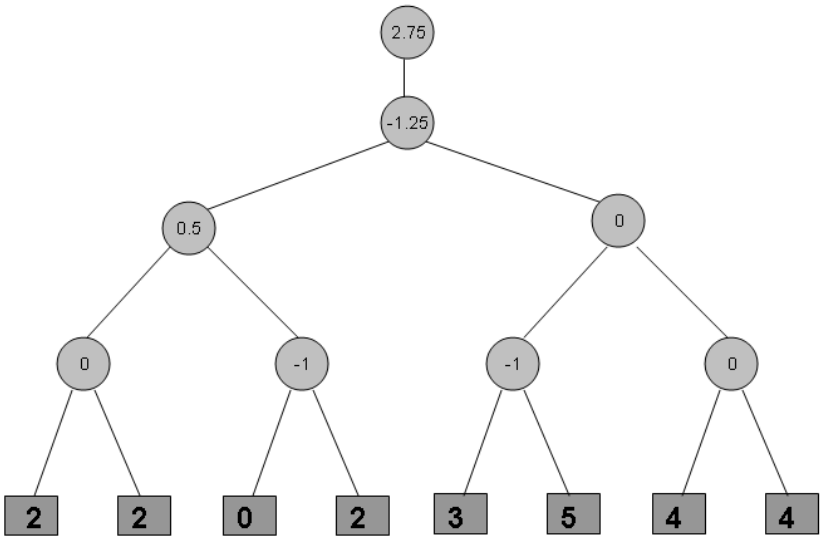

Figure 1: Error tree for $N = 8$

In this paper the purpose is to minimize the L_∞ norm of the vector of absolute or relative errors over all data values. These norms are called the *maximum-absolute error* and the *maximum-relative error*. The question is what M wavelet coefficients should be retained as the wavelet synopsis so as to minimize these error metrics, and how to choose them efficiently.

3 The GK thresholding algorithm

The thresholding algorithm presented in [4] (the "GK" algorithm) is based on a dynamic programming (DP) formulation. Suppose we want to solve the thresholding problem for a subtree T_j rooted at node c_j (which is the coefficient c_j) in the error tree, and we have a storage allocation of m coefficients. In other words, we want to choose m coefficients to retain as a synopsis of T_j, such that the max (absolute or relative) error over the leaves of T_j (the data values) is minimized with respect to any other synopsis of size m of T_j. A straightforward way to solve this kind of problems would be to try to use recursion. There are two possibilities regarding c_j: we can either keep it or drop it. In case we keep (resp. drop) it, we have a storage of $m-1$ (resp. m) coefficients to divide among the two subtrees of c_j. Thus, it seems that all we have to do is choosing the best result among $2m+1$ recursive calls on the two subtrees with the following $2m+1$ possible space divisions:
left: 0 right: $m-1$, left: 1 right: $m-2$,... (resp. left: 0 right: m, left: 1 right: $m-1$,...) for the case we keep (resp. drop) c_j in our synopsis, and choose the division that minimizes the maximum of the two max-errors from both subtrees. This is almost the solution.

The problem with the above solution is that when solving the problems for subtrees, the

algorithm must know what coefficients were selected on the path from c_j to the root (denoted as $path(c_j)$), since different selections along this path would imply different solutions for the sub-trees. We illustrate it using a simple example. Suppose all the coefficients on $path(c_j)$ were retained. Suppose T_j has four leaves, and the errors over its leaves for a given synopsis of size m are $1, 1, 0, 1$ so the max-error is 1. Now suppose we drop some coefficient on $path(c_j)$ with value -1. The errors over the leaves can become $2, 2, 1, 2$ with a max-error of 2, thus changing the optimal solution's value for the subtree, with respect to the case all coefficients on the path were retained. Note that in order to know whether to retain or drop c_j we called the problem recursively on subtrees, but in order to solve the problem on sub-trees we need to know whether c_j (as well as any other coefficient in $path(c_j)$) was retained.

The GK solution for this problem is to call the recursion on the subtrees for every possible selection of coefficients on $path(c_j)$. That is, for each possible selection of subset of retained coefficients on $path(c_j)$ we would call the $2m+1$ recursive calls, and choose the best division for a given space allocation *and* a given coefficient selection along the path to the root. There are no more than $2^{\log N} = N$ possible selections of subsets of retained coefficients along $path(c_j)$, since $path(c_j)$ is at most of length $\log N + 1$.

More formally, let M denote the total space budget for the synopsis, and let T_j be the subtree of the error-tree rooted at node c_j, with coeff(T_j) ($data(T_j)$) denoting the set of coefficient (resp. data) values in T_j. Let $sign_{ik}$ be the sign which the coefficient c_k contributes to the ith data value. Finally, let $A[j, m, S]$ denote the optimal (i.e. minimum) value of the maximum error (relative or absolute) among all data values in T_j assuming a synopsis space budget of m coefficients for the T_j subtree, *and* that a subset $S \subset path(c_j)$ (of size at most $min\{M - m, \log N + 1\}$) of proper ancestors of c_j have been selected for the synopsis; that is, assuming a relative-error metric (denoted as $err_i := relErr_i$, for the relative error over the ith data value),

$$A[j, m, s] = \min_{S_j \subseteq \text{coeff}(T_j), |S_j| \leq m} \left\{ \max_{d_i \in data(T_j)} relErr_i \right\}$$

where

$$relErr_i = \frac{|d_i - \sum_{c_k \in path(d_i) \cap (S_j \cup S)} sign_{ik} \cdot c_k|}{\max\{|d_i|, S\}}$$

The case of absolute errors is defined similarly.

Clearly, $A[0, M, \phi]$ gives us the desired optimal error value at the root node of the error tree. (The algorithm first computes the optimal error value, and keeps in each node $A[j, m, S]$ information of whether c_j was selected, and what is the size of the allocation for the left subtree, so we can trace the computation and build the optimal synopsis later.) The computation of $A[j, m, S]$ is done as follows. For the case we keep c_j, the minimum error is

$$\min_{0 \leq m' \leq m} \max \{A[2j, m', S], A[2j + 1, m - m', S]\}$$

Note that S is a legal subset of retained coefficients on $path(c_{2j})$ and $path(c_{2j+1})$.

For the case we drop c_j, the minimum error is

$$\min_{0 \leq m' \leq m-1} \max \{A[2j, m', S \cup \{c_j\}], A[2j + 1, m - 1 - m', S \cup \{c_j\}]\}$$

Note that again, $S \cup \{c_j\}$ is a legal subset of retained coefficients on $path(c_{2j})$ and $path(c_{2j+1})$.

The base case is computed for a leaf (i.e. a data value) in the tree. In the GK algorithm, the wavelet coefficients are indexed $0, ..., N-1$, and the data values are indexed $N, ..., 2N-1$. That is, for $j \geq N$ they denote $c_j := d_{j-N}$. The base case is

$$A[j, 0, S] = \frac{|d_{j-N} - \sum_{c_k \in S} sign_{j-N,k} \cdot c_k|}{r}$$

where $r = 1$ in the case of absolute errors and $r = \max\{|d_{j-N}|, s\}$ in the case of relative error.

Actually, if we want the entry $A[j, m, S]$ to hold the optimal error for all synopses sizes $\leq m$ and not only for sizes $= m$, we must allow computation of base cases of the form $A[j, m, S]$ ($j \geq N$) for *any* $0 \leq m \leq M$, and not only zero.[2] The reason for this can be demonstrated as follows. Suppose there is a synopsis of size $M-1$ that gives better optimal error than all synopses of size M. We can "catch" this synopsis if we allow computations of $A[j, 1, S]$ for $j \geq N$, since the extra allocation size is "absorbed" at the leaf, and is not used by the algorithm. We used this idea in our optimization of the algorithm.

Complexity. The space complexity of the algorithm is the total number of entries that are computed. This number is $2N \cdot (M+1) \cdot 2N = O(N^2M)$, since there are $2N$ nodes, $M+1$ possible space allocations per subtree rooted at the node, and $2^{\log N+1}$ possible selections of subset of retained coefficients along the path to the root. It can be seen in the base-case formula that each leaf entry can be computed in $O(1)$ time, if we keep the accumulated error (from retaining or droping coefficients on the path) as we go down the tree with the recursion. Every other entry is computed by taking the minimum of the $2M+1$ recursive results, so in the straightforward way the algorithm's running time is $O(N^2M^2)$. However, since the optimal error is monotone decreasing function of storage space (recall that $A[j, m, S]$ is the minimum error value for storage space $\leq m$), each entry is computed in $\log M$ steps using a binary search (details are given in [4]). The total running time of the algorithm is thus $O(N^2M \log M)$.

4 The improved implementation

Our main improvement in the new implemenation is to reduce the space complexity to $O(N^2)$ and the running time to $O(N^2 \log M)$, based on the following simple observation. In order to compute the final answer we don't need to compute all $O(N^2M)$ entries that are computed by the original algorithm. For example, if we compute $A[j, m, S]$ and the node c_j is at depth d, then there are only 2^d possible selections of coefficients on $path(c_j)$, instead of $2^{\log N+1} = 2N$. In addition, if the size of T_j, denoted $|\text{coeff}(T_j)|$, is smaller than M, then instead of computing $A[j, m, S]$ entries for every $0 \leq m \leq M$, we just need to compute the entries for $0 \leq m \leq |\text{coeff}(T_j)|$.

While the idea is quite simple, a straightforward implementation that just reduces the number of entries is incorrect. Recall that an entry $A[j, m, S]$ should keep the minimum error value for all synopses of size $\leq m$, for a given c_j and S. The reason that in the original algorithm $A[j, m, S]$ keeps an optimal value for *every* storage size $\leq m$, and not only for storage size m,

[2] This additional constraint was not considered by Garofalakis and Kumar [4].

was that it allows computations of entries where $m > |\text{coeff}(T_j)|$, which eventually results in base-case computations of the form $A[i, m', S]$ for $i \geq N$ and $m' \geq 0$, thus "absorbing" a "surplus" of m' allocation storage (since it is not used by the leaf), so that the final allocation for T_j is smaller than m in these cases (in this case it is $m - m'$).

As denoted above, our optimization does not allow recursive calls with allocation sizes that are larger than the tree size, so we have to solve the problem in a different way. If a recursive call was called with size m that is larger than than coeff(T_j), it "resets" the size to coeff(T_j), and continues the computation as if it was called with $m = \text{coeff}(T_j)$. Since we keep the results in a DP array, each entry is computed only once, so all subsequent calls for node c_j with allocation size $>$ coeff(T_j) (and the same subset S of $path(c_j)$), would be "reset" to coeff(T_j), and would thus just return the value stored in the array.

However, this still does not fully solve the problem. Recall ([4]) that when the DP array is computed, each entry holds the optimal error values along with a `isRetained` field, indicating whether to keep or drop c_j, and a `leftAllot` field, indicating the size allocation for the left subtree that was used in order to achieve the optimal error value (the right allocation is simply $m-$`leftAllot`). This way it traces the optimal synopsis by moving to the proper entries of the array. Since we reset the allocation size when it is larger than the tree size, the `leftAllot` field in the parent does not reflect the real allocation that was used to achieve the optimal solution, so in the reconstruction phase it is not known what are the correct entries that should be checked.

In order to solve that, we use the same method while reconstructing the synopses from the DP array. The original algorithm uses the $A[j, m, S]$.`leftAllot` field (we denote it next as `leftAllot`), in order to trace the computation and move to the entries $A[2j,$ `leftAllot`$, ..]$ and $A[2j+1, m -$ `leftAllot`$, ..]$. Since during our computation we reset allocation sizes larger than $|\text{coeff}(T_j)|$ to $|\text{coeff}(T_j)|$, we do it again during the synopsis reconstruction phase, such that if `leftAllot` is larger than $|\text{coeff}(T_{2j})|$, or `m - leftAllot` is larger than $|\text{coeff}(T_{2j+1})|$ we reset it to their sizes, and thus keeping the correctness of the construction.

The second optimization we mentioned was computing only entries where S is a legal selection of coefficients on $path(c_j)$. That is, S is one of a 2^d possible subsets of d coefficients, where d is the depth of c_j, instead of one of $2^{\log N}$ subsets of $\log N$ coefficients, which is only correct for the leaves. Note that by the recursion definition, it is not possible to call a recursion with illegal S, so all we had to do is to not allocate these entries in the array.

We quantified the number of computations of our optimized algorithm in table 2 (it is recommended to view the table bottom up). For purposes of exposition we assumed M is a power of 2. In case it is not, the computation is done similarly. For each level we count the total number of entries computed for the level, where the leaves are at the bottom row of the table, and the root is in the first row of the table. The total number of entries in the DP array is the sum of the righthand column, which is $O(N^2)$. Since each entry is computed in $\log M$ time, the total running time of the optimized algorithm is $O(N^2 \log M)$.

4.1 An I/O efficient implementation

We obtain an I/O efficient algorithm as follows. we allocate the DP array such that given a c_j and S, all the entries $A[j, m, S]$ where $0 \leq m \leq \text{coeff}(T_j)$ are continuous in memory. This is done

Level	nodes number	possible space allocations	possible path selections	total entries
1	$\frac{N}{N}=1$	M	$\frac{N}{\frac{N}{2}}=2$	$2M$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$l-2$	$\frac{N}{4M}$	M	$\frac{N}{2M}$	$\frac{N^2}{8M}$
$l-1$	$\frac{N}{2M}$	M	$\frac{N}{M}$	$\frac{N^2}{2M}$
l	$\frac{N}{M}$	M	$\frac{N}{\frac{M}{2}}$	$\frac{N^2}{\frac{M}{2}}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\log N-2$	$\frac{N}{4}$	4	$\frac{N}{2}$	$\frac{N^2}{2}$
$\log N-1$	$\frac{N}{2}$	2	N	N^2
$\log N$	N	1	2N	$2N^2$

Table 2: Number of computed entries per level

by letting m be the third parameter while allocating the array (meaning $A[j,S,m]$, instead of $A[j,m,S]$). This way the binary search for computing each entry is done using one I/O, instead of $\log M$ possible random accesses to memory, implying a worst case of $O(\log M)$ I/Os per entry computation.

5 Experiments

In this section we describe our experiments using the GK syopsis. We first present experiments on point queries, measuring both accuracy and running time, and then present experiments on range queries. All experiments were executed using the τ-synopses system [8].

We describe several experiments, each representing a group of experiments we did. Briefly, our experiments demonstrate several facts:

- For *point queries*, the GK achieved significantly lower maximum-error compared to other wavelet synopses.
- For *range-sum queries with uniform workload*, the standard wavelet achieved lower MSE and maximum-absolute error compared to the GK synopsis.
- For *range-sum queries with various workloads*, the adaptive-greedy synopsis achieved significantly lower MRE and maximum-relative errors compared to the GK synopsis.
- When building the GK synopsis (absolute error version) over the prefix-sums of the data, the synopses it builds give very similar approximation errors to the ROPS synopsis [9], which minimizes the MSE for range-sum queries and is built in linear-I/O optimal time. (The ROPS synopsis is almost identical to the standard synopsis). Thus, it may be that in the case of prefix-sums these two methods build almost similar synopses.
- Running time experiments verified the factor M improvement in the running time of the algorithm.

Note. When the synopsis size is very small, the GK relative-error synopsis throws all the coefficients, resulting in an answer of 0 to each query, and thus with a relative error of 1.0. The reason is that when the number of coefficients is very small there are cases where if we keep only few coefficients, an answer to a query can be, for example 250 instead of 100, resulting in a relative error of 2.5, so the better choice is to threshold all the coefficients, and keep a relative error of 1.

We next call the relative error based GK synopsis "GK-rel" and the absolute error based GK synopsis "GK-abs". We call the synopsis based on the greedy heuristic [11, 17] the "standard synopsis". When refering to "data-size", we mean the number of distinct values in the relation.

5.1 Experiments with point queries

In this section we present experiments measuring the accuracy of the GK synopsis and the running time of the GK algorithm.

5.1.1 Running time - GK algorithm vs. the improved implementation

We compare the running time of the GK algorithm with the running time of our optimized version. We also demonstrate that both algorithms find the same synopses. We used a workload of 1000 queries, with synopses sizes 10-100, over two data-sets taken from KDD.

The first data set was of size 512. Our optimized version's running time was 1:23 minutes, which includes building a synopsis and answering 1000 queries, for each synopsis size. The original GK got an "outOfMemoryException" for synopsis size 40, since the space complexity of the GK algorithm caused a memory explosion on a 512 MB memory machine: $O(N^2 M \log M) = O(512^2 40 \log 40)$ floating point numbers, are approximately 400MB of memory. We then compared both algorithm over a KDD data of size 128, with a workload of 1000 queries, synopses sizes 10-40.

Figure 2 depicts the approximation errors of the two synopses for various synopses sizes. It can be seen that the approximation error is the same for both versions, demonstrating that the same synopses are built by both methods.

We measured the synopes-building running time foreach size: 10, 20, 30 and 40. Our measurements verified the factor M improvement in the running times. The results are depicted in Table 3.

Synopsis size	Original GK run-time (sec)	Optimized GK run-time (sec)
10	0.2	2
20	0.5	9
30	0.6	21
40	1.1	40

Table 3: Running times of the GK algorithm vs. our improved implementation

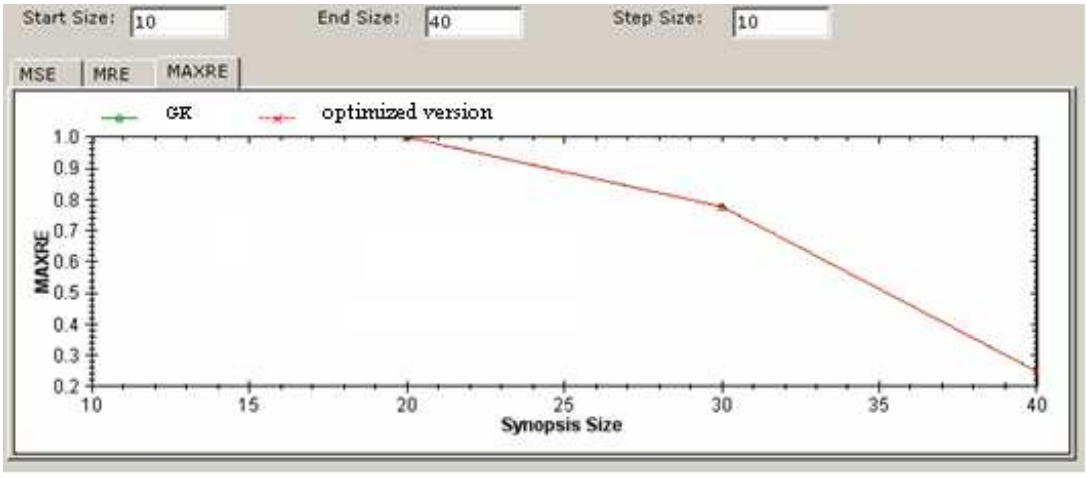

Figure 2: The accuracy of the GK synopses constructed by the GK algorithm and by our improved implementation. As can be seen, both plots coincide, since the constructed synopses are identical. Dataset: KDD data of size 128, workload: 1000 Zipf distributed queries, synopses sizes: 10-40. This figure demonstrates that the results of the improved implementation are the same as the results of the GK implementation. The running time improvement is described in the text.

5.1.2 Max-relative error - the GK synopsis vs. the standard synopsis

In Figure 3 we compared the GK-rel synopsis with the standard synopsis which is optimal for the L_2 norm. We used the KDD data of size 512, with a workload of 1000 queries, with a zipf(0.5) distibution, and synopses sizes of 10-100. The advantage of the GK synopsis is clear (its error is more than 70 times smaller that that of the standard synopsis).

5.1.3 Max-absolute error - the GK synopsis vs. the standard synopsis

In Figure 4 we compared the GK-abs synopsis with the standard synopsis We used the KDD data of size 512, with a workload of 1000 queries, and synopses sizes of 10-100. The advantage of the GK synopsis is smaller than in the relative-error case, as the standard synopsis is aimed at minimizing the MSE. Yet it achieves approximation error which is 1.25-2 times smaller than the standard synopsis

5.2 Experiments on range sum queries

In this section we present experiment on workloads of range sum queries.

5.2.1 Max and mean relative errors over the raw data - the GK-rel synopsis vs. a workload-based synopsis for range-queries

In this section we used workloads of 1000 range-sum queries. We compared the GK-rel synopsis to a workload-based synopsis [7, 13], which is built using an adaptive greedy algorithm that

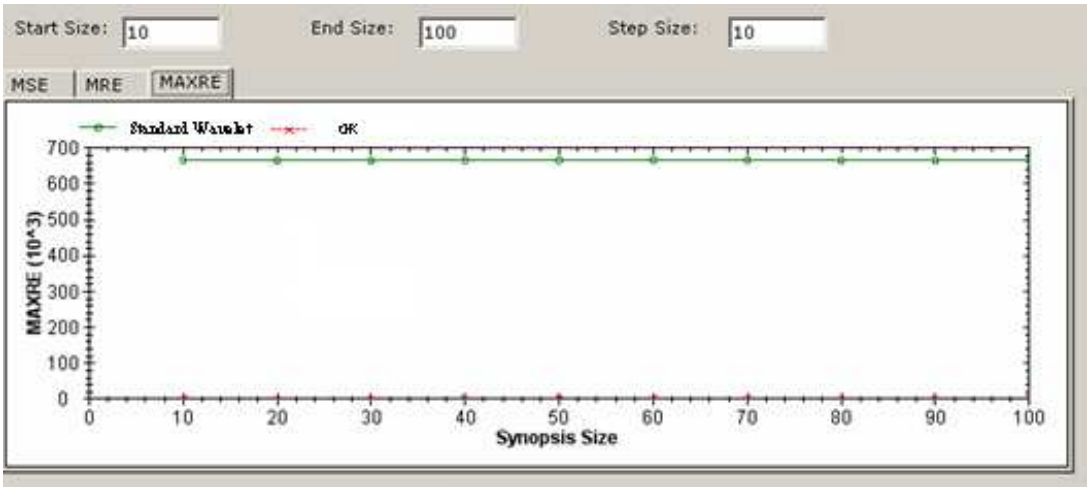

Figure 3: The GK synopsis vs. the standard thresholding. Dataset: KDD data of size 512, workload: 1000 queries, synopses sizes: 10-100. The comparison of two synopses was done with respect to the maximum relative error measure. Since the standard thresholding is designed for minimizing the MSE error measure, the advantage of the GK is clearly seen here, as its plot coincides with the x-axis.

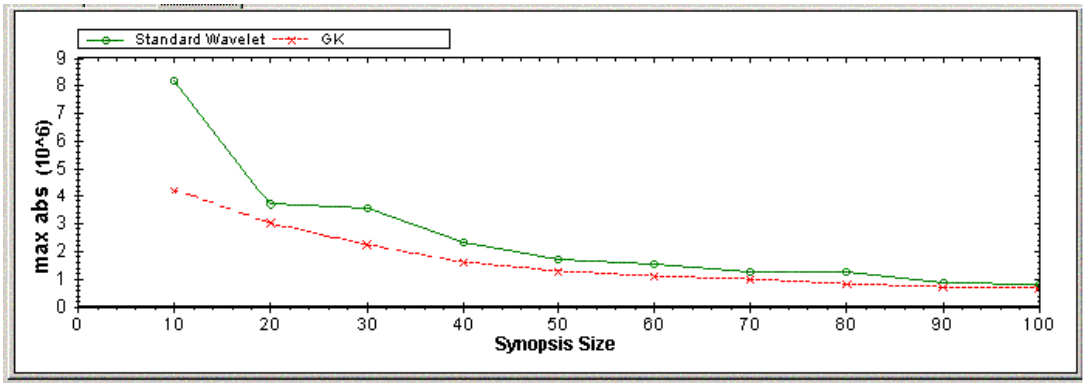

Figure 4: The GK synopsis vs. the standard thresholding. Dataset: KDD data of size 512, workload: 1000 queries, synopses sizes: 10-100. The comparison of two synopses was done with respect to the maximum absolute error measure. The advantage of the GK synopsis can be seen here.

reduces a relative error measured over a workload of range queries. We see in Figures 5 and 6 that the apdaptive greedy algorithm is better with respect to the workload based mean-relative error and even with respect to the max relative error. How can it be? Recall that the GK synopsis minimizes an error measured over *point queries* while the adaptive greedy minimizes an error measured over range-sum queries. The GK synopsis minimizes the max relative error over the data values, which does not necessarily imply minimization of an error measured over range-sum queries.

5.2.2 MSE and max-absolute errors on the raw data - the GK synopsis vs. the standard synopsis

Here we compared the GK-abs algorithm with the standard wavelet synopsis over a uniform workload of range queries. The standard synopsis was superior to the GK-abs synopsis with respect to both maximum-absolute error measure and the MSE measure. We used the KDD data of size 512, with a workload of 1000 queries, and synopses sizes of 10-100. In Figure 7 we present the MSE experiment and in Figure 8 we present the maximum-absolute error experiment.

5.2.3 Max-absolute error on prefix sums - the GK synopsis vs. the ROPS synopsis

We considered the application of the GK synopsis for range-sum queries and obtained a rather surprising result. We used the GK synopsis to answer range sum queries using the method introduced in [11, 17], a method that can be used for any wavelet synopsis. The wavelet synopsis was built over the prefix-sums of the data, in which case a range query can be approximated as the difference between the two (approximated) prefix sum values of the two limits of the range of the query. We compared this implementation to the ROPS synopsis that we have recently introduced [9], which is also built over the prefix-sums of the data. Generally, each of these two synopses is built for a different purpose. The ROPS synopsis minimizes the mean-squared error

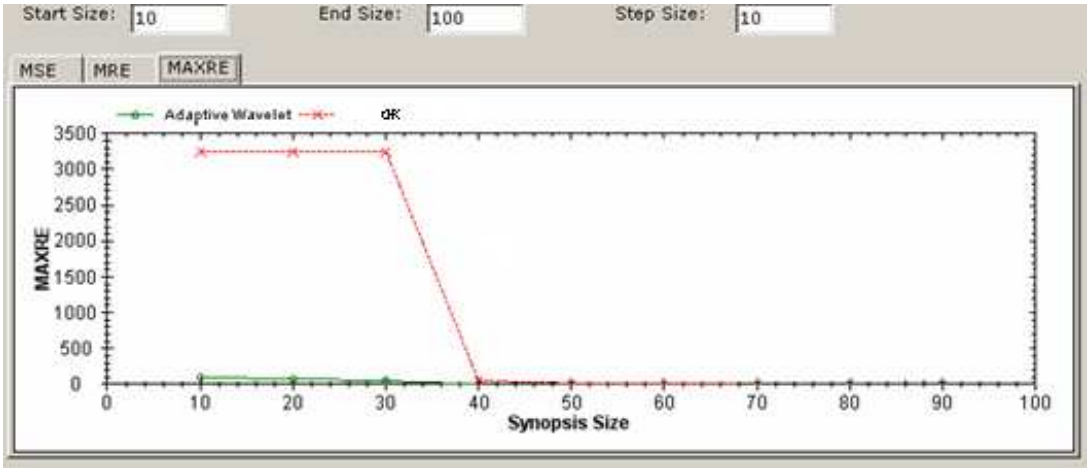

Figure 5: The GK-rel synopsis vs. the workload based synopsis - MaxRE. Dataset: KDD data of size 512, workload: 1000 queries, synopses sizes: 10-100, error measure: max-relative error. The workload-based synopsis achieved significantly smaller errors for small synopses sizes. The GK synopsis is optimal for minimizing the max-relative error measure over all point queries, while here the workload of queries consisted range-sum queries, and therefore the workload-based synopsis achieved better results.

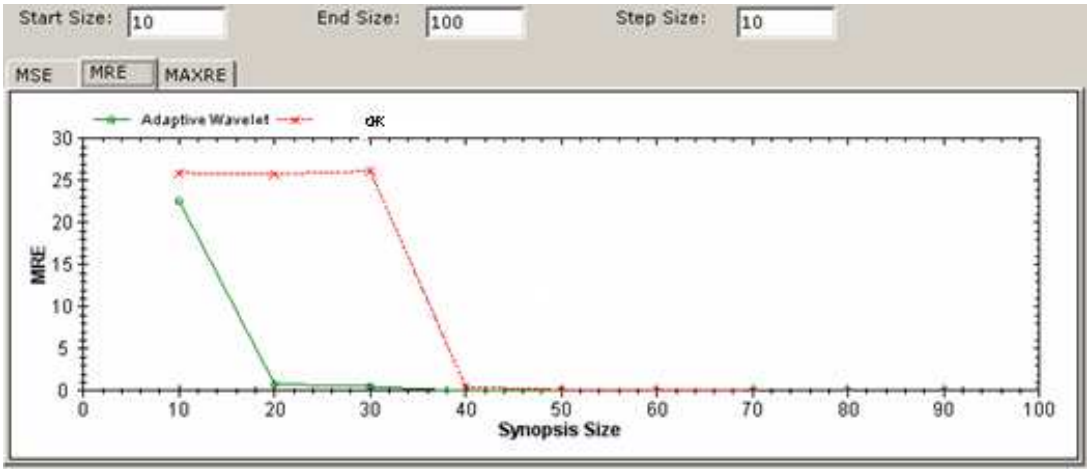

Figure 6: The GK-rel synopsis vs. the workload based synopsis - MRE. Dataset: KDD data of size 512, workload: 1000 queries, synopses sizes: 10-100. The adaptive algorithm achieved better results for small synopses sizes, since it is designed for minimizing the MRE measure for a workload of range queries, while the GK synopsis is designed for minimizing the maximum relative error over all point queries.

over all possible range-sum queries when the transform is done over prefix-sums of the data, and is computed in linear time. The GK synopsis minimizes the maximum absolute error, measure over all data values (in this case over all prefix-sums values). We compared the two synopses

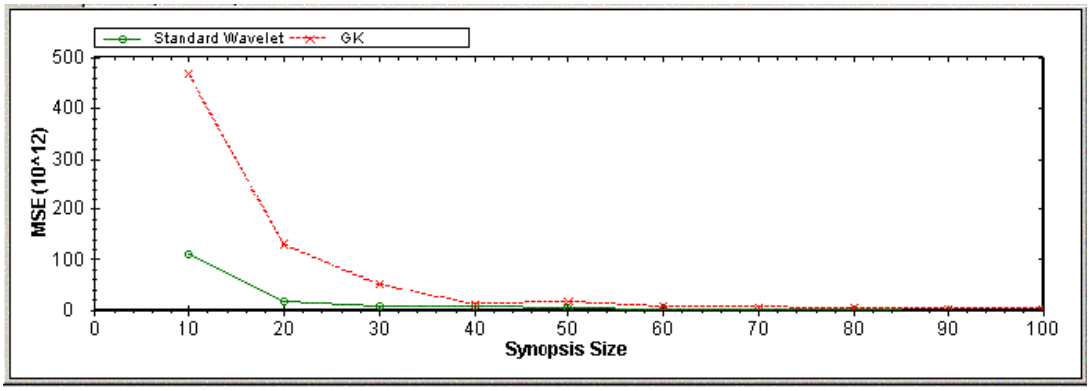

Figure 7: The GK-abs synopsis vs. the standard synopsis. Dataset: KDD data of size 512, workload: 1000 queries, synopses sizes: 10-100. The standard synopsis is superior to the GK synopsis with respect to the MSE measured over the range-queries.

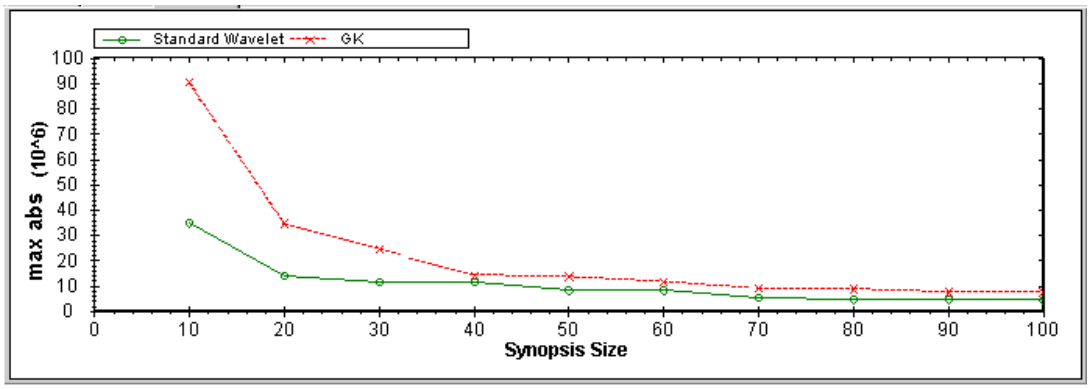

Figure 8: The GK-abs synopsis vs. the standard synopsis. Dataset: KDD data of size 512, workload: 1000 queries, synopses sizes: 10-100. The standard synopsis is superior to the GK synopsis with respect to the maximum-absolute error measured over the range-queries.

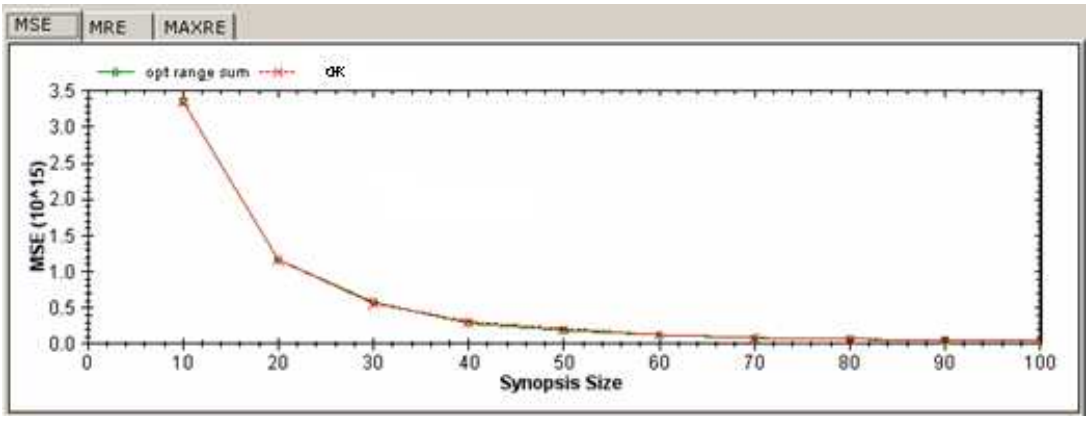

Figure 9: The GK-abs synopsis vs. the ROPS synopsis. Dataset: KDD data of size 512, workload: 1000 queries, synopses sizes: 10-100. It can be seen that for each synopsis size, both synopses achieved the same error with respect to the given workload of queries. We got this result for the maximum-absolute, maximum-relative, MSE and MRE error measures, for each workload that we tried, over each KDE dataset that we had. This hints that both synopses contains very similar subset of coefficients.

over the prefix sums of the data with workloads of 1000 range queries and got exactly the same error value for each workload we tried, over each KDD data-set we have, for both the MSE, MRE, maximum-absolute and maximum-relative error measures. This may suggest that both methods build very similar synopses. The experiment is depicted in Figure 9. We got similar results over different data-sets and workloads.

6 Conclusions

In this paper we provide an improved impelemtnation of the GK algorithm that minimizes the max relative or absolute error over the data values (equivalently on point queries). We improved the running time of the algorithm from $O(N^2 M \log M)$ to $O(N^2 \log M)$ and the space complexity from $O(N^2 M)$ to $O(N^2)$. The synopsis shows good results for point queries. When comparing the results for range-queries, the standard wavelet synopsis was superior to the GK-abs synopsis with respect to MSE and maximum-absolute errors, and the adaptive-greedy algorithm was superior the the GK-rel synopsis with respect to the MRE and maximum-relative errors. An open question for future research is whether there is an efficient algorithm that minimizes these error measures. When comparing the GK-abs with the ROPS synopsis, both built over the prefix sums of the data, the results were pretty similar for each data set, workload and synopsis size we tried. It would be interesting to understand the extent to which these two synopses are similar.

We saw that the GK synopsis, which is good for point queries (with respect to relative and absolute errors), and for range-sum queries over prefix-sums (with respect to abs errors) does not necessarily extend to range-sum queries over the raw data, so a future work can be finding an algorithm that minimizes the max error measured over range-sum queries in the raw-data

case, and for relative errors over prefix-sums. Another direction can be to try minimizing the ratio error. We saw that when the synopsis size is too small the GK-rel algorithm throws all the coefficients so the relative error would not be more than 1. This is not very useful, and the ratio error direction could be more correct in these cases.

Acknowledgments: This work was originated from from a project in the class "Algorithms for Massive Data-sets", given by Yossi Matias and Ely Porat at the spring of 2004. We thank Eitan Yaffe, Daniel's project partner, who helped with the implementation of the GK algorithm.

References

[1] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, 2000*, pages 111–122.

[2] A. Deligiannakis and N. Roussopoulos. Extended wavelets for multiple measures. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 229–240.

[3] M. Garofalakis and P. B. Gibbons. Wavelet synopses with error guarantees. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, 2002.

[4] M. Garofalakis and A. Kumar. Deterministic wavelet thresholding for maximum-error metrics. In *Proceedings of the 2004 ACM PODS international conference on on Management of data*, pages 166–176.

[5] P. B. Gibbons and Y. Matias. Synopsis data structures for massive data sets. In *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science: Special Issue on External Memory Algorithms and Visualization, A*, 1999.

[6] Sudipto Guha. A note on wavelet optimization. Technical report, Department of Computer Science, University of Pennsylvania, 2004.

[7] Y. Matias and L. Portman. Workload-based wavelet synopses. Technical report, Department of Computer Science, Tel Aviv University, 2003.

[8] Y. Matias and L. Portman. τ-synopses: a system for run-time management of remote synopses. In *International conference on Extending Database Technology (EDBT), Software Demo, 865-867 & ICDE'04, Software Demo*, March 2004.

[9] Y. Matias and D. Urieli. Optimal wavelet synopses for range-sum queries over prefix-sums. Technical report, Department of Computer Science, Tel-Aviv University, 2004.

[10] Y. Matias and D. Urieli. Optimal workload-based weighted wavelet synopses. In *Proceedings of the 2005 ICDT conference*, Edinburgh, January 2005.

[11] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 448–459, Seattle, WA, June 1998.

[12] S. Muthukrishnan. Nonuniform sparse approximation using haar wavelet basis. Technical report, DIMACS, May 2004.

[13] L. Portman. Workload-based wavelet synopses. Master's thesis, School of Computer Science, Tel Aviv University, 2003.

[14] E. J. Stollnitz, T. D. Derose, and D. H. Salesin. *Wavelets for Computer Graphics*. Morgan Kaufmann, 1996.

[15] J. S. Vitter and M. Wang. Approximate computation of multidimensional aggregates of sparse data using wavelets. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 193–204, Phildelphia, June 1999.

[16] J. S. Vitter, M. Wang, and B. Iyer. Data cube approximation and histograms via wavelets. In *Proceedings of Seventh International Conference on Information and Knowledge Management*, pages 96–104, Washington D.C., November 1998.

[17] M. Wang. *Approximation and Learning Techniques in Database Systems*. PhD thesis, Duke University, 1999.