Probabilistic Wavelet Synopses

MINOS GAROFALAKIS
Bell Labs, Lucent Technologies, Murray Hill, New Jersey
and
PHILLIP B. GIBBONS
Intel Research, Pittsburgh, Pennsylvania

Recent work has demonstrated the effectiveness of the wavelet decomposition in reducing large amounts of data to compact sets of wavelet coefficients (termed "wavelet synopses") that can be used to provide fast and reasonably accurate approximate query answers. A major shortcoming of these existing wavelet techniques is that the quality of the approximate answers they provide varies widely, even for identical queries on nearly identical values in distinct parts of the data. As a result, users have no way of knowing whether a particular approximate answer is highly-accurate or off by many orders of magnitude. In this article, we introduce *Probabilistic Wavelet Synopses*, the first wavelet-based data reduction technique optimized for guaranteed accuracy of individual approximate answers. Whereas previous approaches rely on *deterministic* thresholding for selecting the wavelet coefficients to include in the synopsis, our technique is based on a novel, *probabilistic* thresholding scheme that assigns each coefficient a probability of being included based on its importance to the reconstruction of individual data values, and then flips coins to select the synopsis. We show how our scheme avoids the above pitfalls of deterministic thresholding, providing *unbiased*, highly accurate answers for individual data values in a data vector. We propose several novel optimization algorithms for tuning our probabilistic thresholding scheme to minimize desired error metrics. Experimental results on real-world and synthetic data sets evaluate these algorithms, and demonstrate the effectiveness of our probabilistic wavelet synopses in providing fast, highly accurate answers with improved quality guarantees.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**]: *probabilistic algorithms (including Monte Carlo)*; H.2.4 [**Database Management**]: Systems—*query processing*

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Wavelets, data synopses, approximate query processing, randomized rounding

A preliminary version of this article appeared as GAROFALAKIS, M. AND GIBBONS, P. B. 2002. Wavelet synopses with error guarantees. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data* (Madison, Wisc., June) ACM, New York.
Most of this work was done while P. B. Gibbons was with Bell Labs.
Authors' addresses: M. Garofalakis, Bell Labs, Lucent Technologies, 600 Mountain Avenue, Murray Hill, NJ 07974; email: minos@research.bell-labs.com; P. B. Gibbons, Intel Research, 417 South Craig Street, Pittsburgh, PA 15213; email: phillip.b.gibbons@intel.com.

1. INTRODUCTION

Approximate query processing over precomputed data synopses has emerged as a cost-effective approach for dealing with the huge data volumes, the high query complexities, and the increasingly stringent response-time requirements that characterize today's Decision Support Systems (DSS) applications. Typically, DSS users pose very complex queries to the underlying Database Management System (DBMS) that require complex operations over gigabytes or terabytes of disk-resident data and, thus, take a very long time to execute to completion and produce exact answers. Due to the *exploratory nature* of many DSS applications, there are a number of scenarios in which an exact answer may not be required, and a user may, in fact, prefer a fast, approximate answer. For example, during a drill-down query sequence in ad-hoc data mining, initial queries in the sequence frequently have the sole purpose of determining the truly interesting queries and regions of the database [Hellerstein et al. 1997]. Providing (reasonably accurate) approximate answers to these initial queries gives users the ability to focus their explorations quickly and effectively, without consuming inordinate amounts of valuable system resources. An approximate answer can also provide useful feedback on how well posed a query is, allowing DSS users to make an informed decision on whether they would like to invest more time and resources to execute their query to completion. Moreover, approximate answers obtained from appropriate *synopses* of the data may be the only available option when the base data is remote and unavailable [Amsaleg et al. 1997]. Finally, for DSS queries requesting a numerical answer (e.g., total revenues or annual percentage), it is often the case that the full precision of the exact answer is not needed and the first few digits of precision will suffice (e.g., the leading few digits of a total in the millions or the nearest percentile of a percentage) [Acharya et al. 1999].

Wavelets provide a mathematical tool for the hierarchical decomposition of functions, with a long history of successful applications in signal and image processing [Jawerth and Sweldens 1994; Natsev et al. 1999; Stollnitz et al. 1996]. Recent studies have also demonstrated the applicability of wavelets to selectivity estimation [Matias et al. 1998] and to approximate query processing over massive relational tables [Chakrabarti et al. 2000; Vitter and Wang 1999] and data streams [Matias et al. 2000; Gilbert et al. 2001]. Briefly, the idea is to apply wavelet decomposition to the input relation (attribute column(s) or OLAP cube) to obtain a compact data synopsis that comprises a select small collection of *wavelet coefficients*. The results of Chakrabarti et al. [2000] and Vitter and Wang [1999] have demonstrated that fast and accurate approximate query processing engines can be designed to operate solely over such compact *wavelet synopses*.

1.1 The Problem with Wavelets

A major shortcoming of existing wavelet-based techniques for approximate query processing is that the quality of the answers they provide varies widely, and users have no way of knowing the accuracy of any particular answer. (Coefficients in the synopsis are typically chosen to optimize an overall error metric,

Table I. Errors with Conventional Wavelet Synopses

Original data values	127	71	87	31	59	3	43	99
	100	42	0	58	30	88	72	130
Wavelet answers	65	65	65	65	65	65	65	65
	100	42	0	58	30	88	72	130

e.g., the L^2 error in the approximation [Stollnitz et al. 1996].) In fact, even for the simplest case of approximating a value in the original data set the absolute errors can vary widely (same for the relative errors). Consider the example depicted in Table I. The first line shows the 16 original data values (the exact answer), whereas the second line shows the 16 approximate answers returned when using conventional wavelet synopses and storing 8 coefficients (details are given in Section 3.1). Although the first half of the values is basically a mirror image of the second half, all the approximate answers for the first half are 65, whereas all the approximate answers for the second half are exact! Similar data values have widely different approximations, for example, 30 and 31 have approximations 30 and 65, respectively. The approximate answers make the first half appear as a uniform distribution, with widely different values, for example, 3 and 127, having the same approximate answer 65. Moreover, the results do not improve when one considers the presumably easier problem of approximating the sum over a range of values: for *all possible* ranges within the first half involving $x = 2$ to 7 of the values, the approximate answer will be $65 \cdot x$, while the actual answers vary widely. For example, for both the range d0 to d2 and the range d3 to d5, the approximate answer is 195, while the actual answer is 285 and 93, respectively. On the other hand, *exact* answers are provided for all possible ranges within the second half (although the user will not know this).

Our example illustrates the following serious problems for approximate query processing with wavelet synopses, resulting from their deterministic approach to selecting coefficients and their focus on overall error metrics instead of the quality of individual answers:

(1) The quality of the answers can vary widely, even
 —within the same data set,
 —when the range is the same width, and is large (e.g., almost half the range),
 —when the synopsis is large (e.g., half the data vector size),
 —when the data values in the range are nearly the same, and
 —when the actual answer is the same or nearly the same.

 These are circumstances where one might expect approximate answers of similar quality.
(2) There are no informative guarantees on the accuracy of a particular answer. As a result, the user has no way of knowing whether a particular answer is highly accurate or off by many orders of magnitude.
(3) Moreover, the approximate answers can systematically favor certain regions of the data or certain values, and there is no bound on this answer bias.

1.2 Our Solution: Probabilistic Wavelet Synopses

In this article, we propose a novel approach to building wavelet synopses that enables *unbiased*,[1] highly accurate approximate query answers with improved guarantees on the quality of individual answers, thereby mitigating the three problems outlined above. Whereas conventional wavelet synopses rely on *deterministic* thresholding for selecting the wavelet coefficients to include in the synopsis, our technique is based on a novel, *probabilistic* thresholding scheme that assigns each coefficient a probability of being included based on its importance to the reconstruction of individual data values, and then flips coins to select the synopsis. Our basic scheme deterministically retains the most important coefficients while randomly rounding the other coefficients either up to a larger value or down to zero. This *randomized rounding* enables unbiased reconstruction of individual data values as well as unbiased answers for any range aggregate query. The basic scheme is contrasted with an alternative scheme we propose in which coefficients are either selected or not (but never rounded up) according to the assigned probabilities, resulting in low-bias answers but often with improved accuracy.

The contributions of this article are as follows:

(1) We provide a quantitative and qualitative demonstration of the unpredictable, widely varying errors arising using conventional wavelet synopses for approximate query answering.
(2) We provide the *first* wavelet-based compression technique optimized for guaranteed-accuracy data reconstruction of a data vector. We prove that our technique provides unbiased data reconstruction and unbiased answers to range aggregate queries.
(3) We provide novel optimization algorithms for tuning our probabilistic thresholding scheme to minimize (a) the expected mean-squared error, and (b) an upper bound on the maximum error in the reconstruction of the data. For reconstruction error, we focus on relative error with a sanity bound (details in Section 3), as this is arguably the most important for approximate query answers. (We can also handle absolute error.)
(4) We present a variation on our scheme that allows for reconstruction bias, but selects coefficient probabilities to minimize this bias, often resulting in improved accuracy.
(5) We describe how our approach for data vectors can be extended for use with multi-dimensional data sets.
(6) We demonstrate the effectiveness of our probabilistic wavelet synopses in providing fast, highly accurate answers with improved quality guarantees, using real-world and synthetic data sets.

[1]The *bias* of a probabilistic estimator $\hat{\Theta}$ for a quantity Θ is $|E[\hat{\Theta}]-\Theta|$, that is, the absolute difference between the expected value of the estimator and the quantity being estimated. Clearly, a low bias is preferable to a high bias. If $E[\hat{\Theta}] = \Theta$, the estimator is *unbiased*.

1.3 Outline

Section 2 presents background material on wavelets. Section 3 gives the details on the shortcomings of previous wavelet synopses, and then describes our probabilistic wavelet synopses. After discussing the general approach, we present our optimization algorithms for tuning our scheme. Experimental results on real-world and synthetic data are presented in Section 4. Section 5 describes extensions for multidimensional data. Section 6 describes related work, Section 7 presents some further discussion on our techniques and, finally, Section 8 outlines our conclusions.

2. WAVELET BASICS

Wavelets are 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 [Stollnitz et al. 1996]. The wavelet decomposition has excellent energy compaction and de-correlation properties, which can be used to effectively generate compact representations that exploit the structure of data. Furthermore, wavelet transforms can generally be computed in linear time.

2.1 One-Dimensional Haar Wavelets

Suppose we are given the one-dimensional data vector A containing the $N = 16$ data values depicted in Table I. The Haar wavelet transform of A can be computed as follows. We first average the values together pairwise to get a new "lower-resolution" representation of the data with the following average values [99, 59, 31, 71, 71, 29, 59, 101]. In other words, the average of the first two values (i.e., 127 and 71) is 99, that of the next two values (i.e., 87 and 31) is 59, and so on. Obviously, some information has been lost in this averaging process. To be able to restore the original values of the data array, we need to store some *detail coefficients*, that capture the missing information. In Haar wavelets, these detail coefficients are simply 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 28 since $99 - 71 = 28$, for the second we again need to store 28 since $59 - 31 = 28$. Note that no information has been lost in this process—it is fairly simple to reconstruct the sixteen values of the original data array from the lower-resolution array containing the eight averages and the eight detail coefficients. Recursively applying the above pairwise averaging and differencing process on the lower-resolution array containing the averages, we get the full decomposition shown in Table II.

The *wavelet transform* (also known as the *wavelet decomposition*) of A is the single coefficient representing the overall average of the data values followed by the detail coefficients in the order of increasing resolution. Thus, the one-dimensional Haar wavelet transform of A is given by $W_A = [65, 0, 14, -15, 20, -20, 21, -21, 28, 28, 28, -28, 29, -29, -29, -29]$. Each entry in W_A is called a *wavelet coefficient*. The main advantage of using W_A instead of the

Table II. Computing the One-Dimensional Haar Wavelet Transform

Resolution	Averages	Detail Coefficients
4	[127, 71, 87, 31, 59, 3, 43, 99, 100, 42, 0, 58, 30, 88, 72, 130]	—
3	[99, 59, 31, 71, 71, 29, 59, 101]	[28, 28, 28, −28, 29, −29, −29, −29]
2	[79, 51, 50, 80]	[20, −20, 21, −21]
1	[65, 65]	[14, −15]
0	[65]	[0]

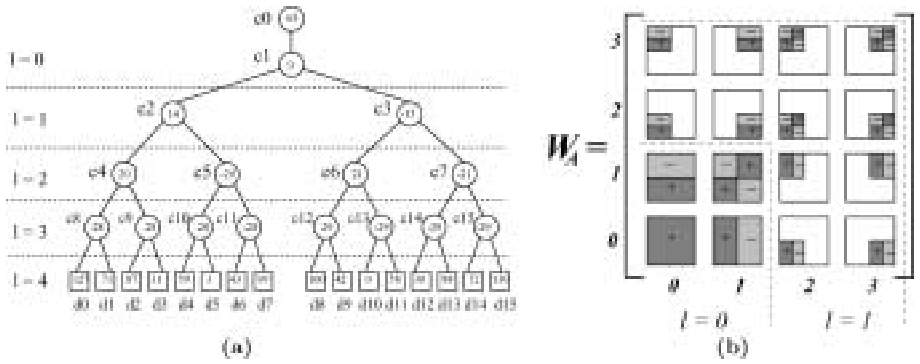


Fig. 1. (a) Error tree structure for our example data array A ($N = 16$). (b) Support regions and signs for the sixteen nonstandard two-dimensional Haar basis functions. The coefficient magnitudes are multiplied by $+1$ (-1) where a sign of $+$ (resp., $-$) appears, and 0 in blank areas.

original data vector A is that for vectors containing similar values most of the detail coefficients tend to have very small values. (However, this is obviously *not* the case with our "bad" example data array.) Thus, eliminating such small coefficients from the wavelet transform (i.e., treating them as zeros) introduces only small errors when reconstructing the original data, giving a very effective form of lossy data compression [Chakrabarti et al. 2000; Stollnitz et al. 1996; Vitter and Wang 1999].

Note that, intuitively, wavelet coefficients carry different weights with respect to their importance in rebuilding the original data values. For example, the overall average is obviously more important than any detail coefficient since it affects the reconstruction of all entries in the data array. In order to equalize the importance of all wavelet coefficients, we need to *normalize* the final entries of W_A appropriately. A common normalization scheme [Stollnitz et al. 1996] is to divide each wavelet coefficient by $\sqrt{2^l}$, where l denotes the *level of resolution* at which the coefficient appears (with $l = 0$ corresponding to the "coarsest" resolution level, as depicted in Table II). Thus, the normalized coefficient, c_i^*, is $c_i / \sqrt{2^{\texttt{level}(c_i)}}$.

2.1.1 *Basic Haar Wavelet Properties and Notational Conventions.* A helpful tool for exploring and understanding the key properties of the Haar wavelet decomposition is the *error tree* structure [Matias et al. 1998]. The error tree is a hierarchical structure built based on the wavelet transform process (even though it is primarily used as a conceptual tool, an error tree can be easily constructed in linear $O(N)$ time). Figure 1 depicts the error tree for our simple

Table III. Notation

Symbol ($i \in \{0..N\text{-}1\}$)	Semantics
A, W_A	Input data array and corresponding wavelet-transform array
N, N_z	Number of cells (nonzero cells, resp.) of array A ($N_z \leq N$)
B	Target number of coefficients retained in a *synopsis* of A ($B < N_z$)
d_i	Data value at cell i of the data array
$\hat{d}_i$	Reconstructed (approximate) data value at cell i based on the synopsis
$d(l : h) = \sum_{i=l}^{h} d_i$	Range sum of data-array values between cells l and h
c_i, c_i^*	Unnormalized (normalized, resp.) Haar coefficient at cell i of the wavelet-transform array
$\texttt{level}(c_i)$	Level of resolution of Haar coefficient c_i
$\texttt{leftleaves}(t)$	Set of leaf nodes in the error subtree rooted at t's left child
$\texttt{rightleaves}(t)$	Set of leaf nodes in the error subtree rooted at t's right child
$\texttt{path}(t)$	Set of all proper ancestor nodes of t in the error tree with nonzero Haar coefficients
T_i	Subtree of the error-tree rooted at the node corresponding to c_i
$\texttt{PATHS}_i$	Set of all root-to-leaf paths in T_i
$E[X]$, $\text{Var}(X)$	Expectation and variance of random variable X
$\texttt{WS}_A$, $\lvert\texttt{WS}_A\rvert$	Probabilistic wavelet synopsis for array A, and the number of retained coefficients
λ_i	Probabilistic rounding value used for coefficient c_i
C_i	Weighted Bernoulli random variable corresponding to coefficient c_i
$y_i = c_i/\lambda_i$	"Success" probability for random variable C_i
k_i, $\text{NORM}(i)$, $\text{NORM}^*(i)$	Normalization terms for L^2 error, relative error, and low bias formulations, respectively
$\texttt{S}$, Δ, $\texttt{q}$	Sanity bound, perturbation value, and quantizing parameter for the relative error algorithm
$\text{NSE}(\hat{d}_i)$	Normalized standard error for the reconstructed data value $\hat{d}_i$
$\text{VAR}(i, y_i)$	Variance of C_i for a given y_i

example data vector A. Each internal node c_i ($i = 0, \ldots, 15$) is associated with a wavelet coefficient value, and each leaf d_i ($i = 0, \ldots, 15$) is associated with a value in the original data array; in both cases, the index i denotes the positions in the (data or wavelet transform) array. For example, c_0 corresponds to the overall average of A. Note that the values associated with the error tree nodes c_j are the *unnormalized* coefficient values; the resolution levels l for the coefficients (corresponding to levels in the tree) are also depicted. We use the terms "node" and "node value" interchangeably in what follows. For ease of reference, Table III summarizes most of the notation used in this article with a brief description of its semantics. Detailed definitions of all these parameters are provided at the appropriate locations in the text. For simplicity, the notation assumes one-dimensional wavelets, as that is the basis for most of the development in this article. Extensions to multidimensional wavelets are straightforward, and any additional notation will be introduced when necessary.

Given an error tree T and an internal node t of T, $t \neq c_0$, we let $\texttt{leftleaves}(t)$ ($\texttt{rightleaves}(t)$) denote the set of leaf (i.e., data) nodes in the subtree rooted at t's left (resp., right) child. Also, given any (internal or leaf) node u, we let $\texttt{path}(u)$ be the set of all (internal) nodes in T that are proper ancestors of u

(i.e., the nodes on the path from u to the root of T, including the root but not u) with nonzero coefficients. Finally, for any two leaf nodes d_l and d_h, we use $d(l : h)$ denote the range sum $\sum_{i=l}^{h} d_i$. Using the error tree representation T, we can outline the following important reconstruction properties of the one-dimensional Haar wavelet decomposition [Matias et al. 1998; Vitter and Wang 1999].

(P1). The reconstruction of any data value d_i depends only on the values of the nodes in $\mathtt{path}(d_i)$. More specifically, we have $d_i = \sum_{c_j \in \mathtt{path}(d_i)} \delta_{ij} \cdot c_j$, where $\delta_{ij} = +1$ if $d_i \in \mathtt{leftleaves}(c_j)$ or $j = 0$, and $\delta_{ij} = -1$, otherwise. For example, in Figure 1, $d_5 = c_0 - c_2 + c_5 - c_{10} = 65 - 14 + (-20) - 28 = 3$. (Note: Because $c_1 = 0$, it is ignored in $\mathtt{path}(d_5)$.)

(P2). An internal node c_j contributes to the range sum $d(l : h)$ only if $c_j \in \mathtt{path}(d_l) \cup \mathtt{path}(d_h)$. More specifically, $d(l : h) = \sum_{c_j \in \mathtt{path}(d_l) \cup \mathtt{path}(d_h)} x_j$, where

$$x_j = \begin{cases} (h - l + 1) \cdot c_j, & \text{if } j = 0 \\ (|\mathtt{leftleaves}(c_j, l : h)| - |\mathtt{rightleaves}(c_j, l : h)|) \cdot c_j, & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathtt{leftleaves}(c_j, l : h) = \mathtt{leftleaves}(c_j) \cap \{d_l, d_{l+1}, \ldots, d_h\}$ (i.e., the intersection of $\mathtt{leftleaves}(c_j)$ with the summation range) and $\mathtt{rightleaves}(c_j, l : h)$ is defined similarly. For example, in Figure 1, $d(3 : 5) = 3c_0 - c_2 - c_4 + 2c_5 - c_9 + (1-1)c_{10} = 195 - 14 - 20 - 40 - 28 + 0 = 93$.

Thus, reconstructing a single data value involves summing at most $\log N + 1$ coefficients and reconstructing a range sum involves summing at most $2 \log N + 1$ coefficients, regardless of the width of the range.

2.2 Multidimensional Haar Wavelets

The Haar wavelet decomposition can be extended to *multidimensional* data arrays using two distinct methods, namely the *standard* and *nonstandard* Haar decomposition [Stollnitz et al. 1996]. Each of these transforms results from a natural generalization of the one-dimensional decomposition process described above, and both have been used in a wide variety of applications, including approximate query answering over high-dimensional DSS data sets [Chakrabarti et al. 2000; Vitter and Wang 1999].

As in the one-dimensional case, the Haar decomposition of a d-dimensional data array A results in a d-dimensional wavelet-coefficient array W_A with the same dimension ranges and number of entries. (The full details as well as efficient decomposition algorithms can be found in Chakrabarti et al. [2000] and Vitter and Wang [1999].) Consider a d-dimensional wavelet coefficient W in the (standard or nonstandard) wavelet-coefficient array W_A. W contributes to the reconstruction of a d-dimensional rectangular region of cells in the original data array A (i.e., W's *support region*). Further, the sign of W's contribution ($+W$ or $-W$) can vary along the quadrants of W's support region in A. As an example, Figure 1(b) depicts the support regions and signs of the sixteen nonstandard, two-dimensional Haar coefficients in the corresponding locations of a 4×4 wavelet-coefficient array W_A. The blank areas for each coefficient correspond to regions of A whose reconstruction is independent of the coefficient, that is, the coefficient's contribution is 0. Thus, $W_A[0, 0]$ is the overall average that contributes positively (i.e.,"$+W_A[0, 0]$") to the reconstruction of all values in A, whereas $W_A[3, 3]$ is a detail coefficient that contributes (with the signs shown in Figure 1(b)) only to values in A's upper right quadrant. Each data cell in A can be accurately reconstructed by adding up the contributions (with the appropriate signs) of those coefficients whose support regions include the cell. Figure 1(b) also depicts the two *levels of resolution* ($l = 0, 1$) for our example two-dimensional Haar coefficients; as in the one-dimensional case, these levels define the appropriate constants for normalizing coefficient values [Chakrabarti et al. 2000; Stollnitz et al. 1996].

Error tree structures for multi-dimensional Haar wavelets can be constructed (once again in linear $O(N)$ time) in a manner similar to those for the one-dimensional case. A major difference is that, in a d-dimensional error tree, each node (except for the root, i.e., the overall average) actually corresponds to a *set* of $2^d - 1$ wavelet coefficients that have the same support region but different quadrant signs for their contribution. Furthermore, each (nonroot) node t in a d-dimensional error tree has 2^d children corresponding to the quadrants of the (common) support region of all coefficients in t. (Note that, given a coefficient W in node t, the sign of W's contribution to the leaf (data) values residing at each of t's children is determined by the quadrant sign information for W.) Returning to our two-dimensional example in Figure 1(b), the (single) child t of the root node in the error tree contains the coefficients $W_A[0, 1]$, $W_A[1, 0]$, and $W_A[1, 1]$, and t has four children corresponding to the four 2×2 quadrants of the array. The child corresponding to the lower-left quadrant contains the coefficients $W_A[0, 2]$, $W_A[2, 0]$, and $W_A[2, 2]$, and all coefficients in t contribute with a "$+$" sign to all values in this quadrant.

Based on the above generalization of the error tree structure to multiple dimensions, we can extend properties (P1) and (P2) to multidimensional Haar wavelets. Our novel technical results and algorithms rely solely on these key properties of Haar wavelets and, therefore, are applicable for general, multidimensional wavelet synopses. However, to simplify the exposition, the development in this article is based primarily on the one-dimensional case; extensions to multidimensional wavelets are discussed in some detail in Section 5.

2.3 Using Wavelets for Data Reduction: Coefficient Thresholding

Given a limited amount of storage for maintaining a *wavelet synopsis* of a data array A, we can only retain a certain number B of the coefficients stored in W_A. (The remaining coefficients are implicitly set to 0.) Letting N_z denote the number of non-zero entries in the data array A, we typically have $B \ll N_z$; that is, the chosen B wavelet coefficients form a highly compressed approximate representation of the original data. The goal of coefficient thresholding is to determine the "best" subset of B coefficients to retain, so that some overall error measure in the approximation is minimized. Conventional coefficient thresholding is a completely deterministic process that typically retains

Table IV. Conventional Wavelet Synopsis Using Deterministic Thresholding

Index i	0	1	2	3	4	5	6	7
	8	9	10	11	12	13	14	15
Wavelet coefficient c_i	65	0	14	-15	20	-20	21	-21
	28	28	28	-28	29	-29	-29	-29
`level`(c_i)	0	0	1	1	2	2	2	2
	3	3	3	3	3	3	3	3
Normalized coefficient c_i^*	65	0	$\frac{14}{\sqrt{2}}$	$\frac{-15}{\sqrt{2}}$	10	-10	$\frac{21}{2}$	$\frac{-21}{2}$
	$\frac{14}{\sqrt{2}}$	$\frac{14}{\sqrt{2}}$	$\frac{14}{\sqrt{2}}$	$\frac{-14}{\sqrt{2}}$	$\frac{29}{2\sqrt{2}}$	$\frac{-29}{2\sqrt{2}}$	$\frac{-29}{2\sqrt{2}}$	$\frac{-29}{2\sqrt{2}}$
Retained coefficient	65	0	0	-15	0	0	21	-21
	0	0	0	0	29	-29	-29	-29

the B largest wavelet coefficients in *absolute normalized value* (an example is given in the next section). It is a well-known fact that, for Haar wavelets, this thresholding method is in fact *provably optimal* with respect to minimizing the overall root-mean-squared error (i.e., L^2-*norm average error*) in the data compression [Stollnitz et al. 1996]. More formally, letting $\hat{d}_i$ denote the (approximate) reconstructed data value for cell i, retaining the B largest normalized coefficients implies that the resulting synopsis minimizes the quantity $\sqrt{\frac{1}{N}\sum_i (d_i - \hat{d}_i)^2}$ (for the given amount of space B).

3. PROBABILISTIC WAVELET SYNOPSES

In this section, we first detail the problems with conventional wavelet synopses, and then present our probabilistic approach based on randomized rounding. We present four different schemes for selecting rounding values for Haar coefficients. Finally, we summarize our approach with an example.

3.1 The Problem with Conventional Wavelet Synopses

As discussed above, conventional wavelet synopses retain the B wavelet coefficients with the largest absolute value after normalization (according to level); this deterministic process minimizes the overall L^2 error in reconstructing all the data values. Unfortunately, these guarantees on overall error do not guarantee the approximation quality of the individual data values or the results of individual range-sum queries.

Our running example discussed earlier in this article (Table I, Table II, Figure 1) illustrates this failing. Table IV depicts the wavelet coefficients in the wavelet transform (W_A) for our example data array A, followed by the level of each coefficient (`level`(c_i)) and the normalized coefficients (c_i^*). With conventional wavelet synopses, we retain the B coefficients c_i with largest $|c_i^*|$. In this example, $|c_0^*|$, $|c_3^*|$, $|c_6^*|$, $|c_7^*|$, $|c_{12}^*|$, $|c_{13}^*|$, $|c_{14}^*|$, and $|c_{15}^*|$ are all greater than 10, while the rest are at most 10. Thus, for $B = 8$, the conventional wavelet synopsis is $c_0, c_3, c_6, c_7, c_{12}, c_{13}, c_{14}$, and c_{15}, as shown in the table.

For a given wavelet synopsis, approximate answers are obtained by assuming all nonretained coefficients are zero, and either applying property (P1) from Section 2.1 to estimate individual values or applying property (P2) from

Table V. Errors with Conventional Wavelets

Original data values	124M	68M	84M	28M	56M	0	40M	96M
	101M	43M	1M	59M	31M	89M	73M	131M
Wavelet transform	64M	−2M	14M	−15M	20M	−20M	21M	−21M
	28M	28M	28M	−28M	29M	−29M	−29M	−29M
Wavelet synopsis	64M	0	0	−15M	0	0	21M	−21M
	0	0	0	0	29M	−29M	−29M	−29M
Wavelet answers	64M	64M	64M	64M	64M	64M	64M	64M
	99M	41M	−1M	57M	29M	87M	71M	129M

Section 2.1 to estimate range sums. For the wavelet synopsis above, Table I depicts the (error-prone) individual values estimates obtained. For example, $d_5 = c_0 - c_2 + c_5 - c_{10}$, and so $\hat{d}_5 = 65 - 0 + 0 - 0 = 65$; since $d_5 = 3$, the estimate has over 2,000% relative error! Likewise, $d(3:5) = 3c_0 - c_2 - c_4 + 2c_5 - c_9$, and so the estimate for this range sum is $195 - 0 - 0 + 0 - 0 = 195$; since $d(3:5) = 93$, the estimate has over a 100% relative error!

The reader may verify that each of the three problems outlined in Section 1 occur in this example, when using a conventional wavelet synopsis. For example, even when the synopsis happens to produce an exact answer to a range sum query that involves the right half of the data values, there is no way of knowing this from the synopsis, because c_1 is not retained.

Moreover, it is not difficult to construct examples with arbitrarily large relative and absolute error. For example, a simple linear transformation of our example data array in Table I yields the example depicted in Table V. Note that the data value d_5 is 0, but its estimate using a conventional wavelet synopsis with $B = 8$ is 64 million, the same as the estimate for $d_0 = 124$ million.

3.1.1 *Root Causes.*

As can be seen from these examples, conventional wavelet synopses suffer from (1) strict deterministic thresholding (i.e., 100% above the threshold are retained, and 0% below the threshold are retained), (2) independent thresholding (i.e., there is no attempt to adapt the thresholding based on what is happening to neighboring coefficients, in order to avoid large regions with no retained coefficients), and (3) the errors resulting from dropping coefficients without compensating for their absence. For example, the retained coefficients (from the right half of the tree in Figure 1) are only slightly larger than the nonretained coefficients (from the left half), and yet *all* the right-half coefficients are retained while *none* of the left-half coefficients are retained, and there is no attempt to compensate for the significant errors that result.

Our approach, outlined next, is to address these three root causes of wavelet synopsis errors, by devising a scheme based on *randomized rounding*,[2] using carefully chosen rounding values.

[2]Randomized rounding [Motwani and Raghavan 1995] has been used previously as a technique for obtaining approximate solutions to integer programs [Raghavan and Thompson 1987] and approximate sums of integers in parallel [Matias 1992]. Our application of randomized rounding is in a completely different domain, requiring new techniques.

3.2 Probabilistic Wavelet Synopses: The General Approach

We seek to overcome the problems with conventional wavelet synopses outlined thus far by introducing a new approach for building wavelet synopses from wavelet-transform arrays. In a nutshell, our scheme deterministically retains the most important coefficients while randomly rounding the other coefficients either up to a larger value (called a *rounding value*) or down to zero. The probability of rounding up vs. down is selected so that the expected value of the rounded coefficient equals the original coefficient. By carefully selecting the rounding values, we ensure that (1) we expect a total of B coefficients to be retained, and (2) we minimize a desired error metric in the reconstruction of the data, for example, the maximum relative error.

The key idea in our thresholding scheme is to associate, with each non-zero coefficient c_i in the wavelet-transform of A, a random variable C_i such that (1) C_i takes the value zero (i.e., c_i is discarded from the synopsis) with some (possibly zero) probability, and (2) $E[C_i] = c_i$. Then $\mathtt{WS}_A$, the probabilistic wavelet synopsis for A, is comprised of the values for those random variables C_i with nonzero values. We determine the general form of these random variables using a randomized rounding scheme, where we select a *rounding value*, λ_i, for each nonzero c_i such that $C_i \in \{0, \lambda_i\}$, $0 < \frac{c_i}{\lambda_i} \leq 1$, and

$$C_i = \begin{cases} \lambda_i & \text{with probability } \frac{c_i}{\lambda_i} \\ 0 & \text{with probability } 1 - \frac{c_i}{\lambda_i}. \end{cases}$$

Thus, our proposed thresholding scheme essentially "rounds" each nonzero wavelet coefficient c_i *independently* to either λ_i or zero by flipping a biased coin with success probability $\frac{c_i}{\lambda_i}$. It is easy to see that for this rounding process the expected value of each rounded coefficient is $E[C_i] = \lambda_i \cdot \frac{c_i}{\lambda_i} + 0 \cdot (1 - \frac{c_i}{\lambda_i}) = c_i$ (i.e., the actual coefficient value), and its variance is simply

$$\mathrm{Var}(C_i) = E[C_i^2] - (E[C_i])^2 = \lambda_i^2 \cdot \frac{c_i}{\lambda_i} - c_i^2 = (\lambda_i - c_i) \cdot c_i. \qquad (2)$$

For the special case where we deterministically retain the coefficient, we set $\lambda_i = c_i$, and indeed $\mathrm{Var}(C_i) = 0$.

3.2.1 *Unbiased Estimation.* Let $\hat{d}_i$ denote the estimate for the data value d_i, as calculated based on the coefficient values retained in our probabilistic wavelet synopsis $\mathtt{WS}_A$, using property (P1) above. Moreover, let $\hat{d}(l : h)$ ($\hat{d}_{avg}(l : h)$) denote the estimate for the range sum $d(l : h)$ (the range average $d(l : h)/(h - l + 1)$, respectively), as calculated based on the coefficient values retained in $\mathtt{WS}_A$, using property (P2) above. Clearly, $\hat{d}_i$, $\hat{d}(l : h)$, and $\hat{d}_{avg}(l : h)$ are random variables. The next theorem shows that these estimators are unbiased.

THEOREM 3.1. *Each of $\hat{d}_i$, $\hat{d}(l : h)$, and $\hat{d}_{avg}(l : h)$ are unbiased estimators for the data value d_i, the range sum $d(l : h)$, and the range average $d(l : h)/(h-l+1)$, respectively.*

PROOF. By property (P1) and the linearity of expectation, we have:

$$E[\hat{d}_i] = E\left[\sum_{c_j \in \mathtt{path}(d_i)} \delta_{ij} \cdot C_j\right] = \sum_{c_j \in \mathtt{path}(d_i)} \delta_{ij} \cdot E[C_j] = \sum_{c_j \in \mathtt{path}(d_i)} \delta_{ij} \cdot c_j = d_i. \quad (3)$$

For each nonzero coefficient c_j, let $X_j = (h - l + 1) \cdot C_j$, if $j = 0$, and $(|\mathtt{leftleaves}(c_j, l : h)| - |\mathtt{rightleaves}(c_j, l : h)|) \cdot C_j$, otherwise. Note that $E[X_j] = x_j$, where x_j is defined in Eq. (1). By property (P2) and the linearity of expectation, we have:

$$\begin{aligned} E[\hat{d}(l : h)] &= E\left[\sum_{c_j \in \mathtt{path}(d_l) \cup \mathtt{path}(d_h)} X_j\right] = \sum_{c_j \in \mathtt{path}(d_l) \cup \mathtt{path}(d_h)} E[X_j] \\ &= \sum_{c_j \in \mathtt{path}(d_l) \cup \mathtt{path}(d_h)} x_j = d(l : h). \end{aligned} \quad (4)$$

The result for $\hat{d}_{avg}(l : h)$ follows immediately from Eq. (4). □

For example, suppose we select $\lambda_0 = c_0$, $\lambda_{10} = 2 \cdot c_{10}$, and $\lambda_i = \frac{3c_i}{2}$ for all other nonzero coefficients c_i in Figure 1. Let $y_i = \frac{c_i}{\lambda_i}$ be the probability of rounding up. Then $E[\hat{d}_5] = E[C_0] - E[C_2] + E[C_5] - E[C_{10}] = y_0\lambda_0 - y_2\lambda_2 + y_5\lambda_5 - y_{10}\lambda_{10} = 65 - \frac{2}{3} \cdot 21 + \frac{2}{3} \cdot (-30) - \frac{1}{2} \cdot 56 = 65 - 14 - 20 - 28 = 3$, which is exactly d_3. Likewise, $E[\hat{d}(3 : 5)] = 3 \cdot E[C_0] - E[C_2] - E[C_4] + 2 \cdot E[C_5] - E[C_9] = 3 \cdot 65 - \frac{2}{3} \cdot 21 - \frac{2}{3} \cdot 30 + 2 \cdot \frac{2}{3} \cdot (-30) - \frac{2}{3} \cdot 42 = 195 - 14 - 20 - 40 - 28 = 93$, which is exactly $d(3 : 5)$.

3.2.2 *The Impact of the λ_i's.* Everything thus far holds for *any* choice of λ_i's, as long as $0 < \frac{c_i}{\lambda_i} \le 1$. The choice of the λ_i's is crucial, however, because it determines the variances of our estimators as well as the expected number of coefficients retained. Indeed, the key to providing "good" error guarantees for individual data values (for range sums) lies in selecting the λ_i's to ensure small variances $\mathrm{Var}(\hat{d}_j)$ of the reconstructed data values (data paths, respectively) while not exceeding the prescribed space limit for the synopsis. Because each coefficient is rounded independently, we have by Eq. (2):

$$\mathrm{Var}(\hat{d}_j) = \mathrm{Var}\left(\sum_{c_i \in \mathtt{path}(d_j)} \delta_{ji} \cdot C_i\right) = \sum_{c_i \in \mathtt{path}(d_j)} (\delta_{ji})^2 \cdot \mathrm{Var}(C_i) = \sum_{c_i \in \mathtt{path}(d_j)} (\lambda_i - c_i) \cdot c_i. \quad (5)$$

Thus, having a λ_i closer to c_i reduces the variance. On the other hand, we retain all nonzero coefficients after the rounding step, and $|\mathtt{WS}_A|$, the number of nonzero coefficients after rounding, is a random variable such that:

$$E[|\mathtt{WS}_A|] = \sum_{i | c_i \ne 0} \frac{c_i}{\lambda_i}. \quad (6)$$

Thus, having λ_i's further from their respective c_i's reduces the expected number of retained coefficients. For a given target B on the number of retained coefficients, our choice of λ_i's needs to ensure that $E[|\mathtt{WS}_A|] \le B$.

3.3 Rounding to Minimize the Expected Mean-Squared Error

A reasonable approach is to select the λ_i values in a way that minimizes some overall error metric (e.g., L^2) in the approximation. Because such error metrics are obviously random variables under our probabilistic methodology, we seek to minimize their *expectation*. The theorem below follows from the orthogonality properties of the Haar basis.

THEOREM 3.2. *For any choice of λ_i's for the non-zero c_i's such that $0 < \frac{c_i}{\lambda_i} \leq 1$, the expected value of the overall L^2 error in reconstructing the data values from a probabilistic wavelet synopsis, $E[L^2] = E[\sum_j (\hat{d}_j - d_j)^2]$, is $\sum_{i|c_i \neq 0} \frac{\mathrm{Var}(C_i)}{2^{\mathtt{level}(c_i)}} = \sum_{i|c_i \neq 0} \frac{(\lambda_i - c_i)\cdot c_i}{2^{\mathtt{level}(c_i)}}$.*

PROOF. Let A denote the original data vector (i.e., the d_j's) and let $\hat{A}$ be the reconstructed, approximate data vector (i.e., the $\hat{d}_j$'s) from a probabilistic wavelet synopsis, as described above. Also, let c_i^* denote the normalized values of the original coefficients and let C_i^* denote the corresponding (normalized) random variables for the probabilistic synopsis ($i = 1, \ldots, N$). Finally, let $\mathbf{u}_i$ denote the orthonormal Haar basis functions for the decomposition, so that the inner product $\langle \mathbf{u}_i | \mathbf{u}_j \rangle = \delta_{ij} = 1$, iff $i = j$, and 0 otherwise [Stollnitz et al. 1996]. Then, by the wavelet transform, we have $A = \sum_{i=1}^{N} c_i^* \mathbf{u}_i$ and $\hat{A} = \sum_{i=1}^{N} C_i^* \mathbf{u}_i$ (where, $\hat{A}$ is obviously a *random* vector). Furthermore, by our probabilistic coefficient construction, we have $E[C_i^*] = c_i^* = c_i / \sqrt{2^{\mathtt{level}(c_i)}}$. Thus,

$$E[L^2] = E\left[\sum_j (\hat{d}_j - d_j)^2\right] = E[\| A - \hat{A} \|^2] = E\,[\langle\, A - \hat{A} | A - \hat{A} \rangle]$$

$$= E\left[\left\langle \sum_{i=1}^{N} (c_i^* - C_i^*)\mathbf{u}_i \,\Big|\, \sum_{j=1}^{N} (c_j^* - C_j^*)\mathbf{u}_j \right\rangle\right],$$

which, by the orthonormality of the u_i's and linearity of expectation, gives:

$$E[L^2] = E\left[\sum_{i=1}^{N} (c_i^* - C_i^*)^2 \langle \mathbf{u}_i | \mathbf{u}_i \rangle\right] = E\left[\sum_{i=1}^{N} (c_i^* - C_i^*)^2\right] = \sum_{i|c_i \neq 0} E[(c_i^* - C_i^*)^2].$$

And, since $E[C_i^*] = c_i^*$ and

$$E[(C_i^*)^2] = \frac{c_i}{\lambda_i} \left(\frac{\lambda_i}{\sqrt{2^{\mathtt{level}(c_i)}}} \right)^2 = \frac{\lambda_i c_i}{2^{\mathtt{level}(c_i)}},$$

we have:

$$E[L^2] = \sum_{i|c_i \neq 0} (E[(C_i^*)^2] - (c_i^*)^2) = \sum_{i|c_i \neq 0} \left(\frac{\lambda_i c_i}{2^{\mathtt{level}(c_i)}} - \frac{c_i^2}{2^{\mathtt{level}(c_i)}} \right)$$

$$= \sum_{i|c_i \neq 0} \frac{(\lambda_i - c_i) c_i}{2^{\mathtt{level}(c_i)}}. \qquad \square$$

Theorem 3.2 shows that the variance of the nonzero coefficients at lower (i.e., coarser) levels of resolution (that is, closer to the root of the error tree) has a higher impact on the overall L^2 error. This is a very intuitive result since, by virtue of the Haar decomposition, such coefficients contribute to the reconstruction of a larger number of data values. Based on Theorem 3.2, selecting the rounding values λ_i to minimize the expected L^2 error subject to a given expected space constraint[3] B can be formally stated as the following optimization problem:

[Expected L^2 Error Minimization]. Find the rounding values λ_i that *minimize* the expected L^2 error $\sum_{i|c_i \neq 0} \frac{(\lambda_i - c_i) \cdot c_i}{2^{\texttt{level}(c_i)}}$, subject to the constraints $0 < \frac{c_i}{\lambda_i} \leq 1$ for all nonzero c_i and $E[|\texttt{WS}_A|] = \sum_{i|c_i \neq 0} \frac{c_i}{\lambda_i} \leq B$. □

3.3.1 *An Optimal Algorithm for Computing the λ_i Values.* The above-stated problem is a continuous, nonlinear optimization problem with an objective function that is *convex* in the problem variables. In general, such *convex programming* problems are solved using computationally intensive numerical methods (e.g., Sequential Quadratic Programming (SQP) or interior-point methods), that are typically not intended to scale beyond a few thousand variables. The basic assumption is that N is small enough for all of the input data to be resident in main memory [D. M. Gay, Personal communication 2001]. Such assumptions are clearly unrealistic when dealing with data-reduction problems for large, possibly multidimensional, databases; the number of data cells N can often be in the order of several millions or even billions.

Fortunately, the specific form of our L^2 error minimization problem allows us to derive an efficient optimal algorithm for computing the rounding values λ_i. More specifically, letting $y_i = \frac{c_i}{\lambda_i}$ (i.e., the "success" probability for random variable C_i) and $k_i = c_i^2 / 2^{level(c_i)}$, it is easy to see that our expected L^2 error minimization problem is equivalent to:

$$\text{Minimize} \quad \sum_{i|c_i \neq 0} \frac{k_i}{y_i} \quad \text{subject to} \quad \sum_{i|c_i \neq 0} y_i \leq B \quad \text{and} \quad y_i \in (0, 1].$$

(Note that terms involving only c_i's are constant in our minimization problem, and hence safely ignored.) It is not hard to see that using B space is always better than using less than B; thus, at optimality, $\sum_{i|c_i \neq 0} y_i = B$. Then, based on the Cauchy-Schwarz inequality, the minimum value of the objective is reached when $\frac{\sqrt{k_i}}{y_i}$ is the same for all i with $c_i \neq 0$. Let $w = \frac{\sqrt{k_i}}{y_i}$, so that we require $\sum_{i|c_i \neq 0} y_i = \sum_{i|c_i \neq 0} \frac{\sqrt{k_i}}{w} = B$. This gives $w = \frac{1}{B} \sum_i \sqrt{k_i}$ or, equivalently, $y_i = B \cdot \sqrt{k_i} / \sum_i \sqrt{k_i}$, for all i with nonzero c_i. Thus, setting

$$\lambda_i = \frac{c_i}{y_i} = \frac{c_i \cdot \sum_i \sqrt{k_i}}{B \cdot \sqrt{k_i}}, \tag{7}$$

for all i with nonzero c_i, is the optimal solution, ignoring the second constraint that $y_i \in (0, 1]$.

[3]For simplicity, we discuss *expected* space constraints in this paper, although one can ensure we are within an absolute space bound B^*, by targeting an expectation slightly less than B^* and possibly repeating the coin tossing a few times until $|\texttt{WS}_A| \leq B^*$. In our experiments comparing wavelet approaches, we ensure that all synopses have the same number of coefficients.

```
procedure MinL2( $W_A$, $B$ )
Input: Array $W_A = [c_0, \ldots, c_{N-1}]$ of $N$ Haar wavelet coefficients and space budget $B$ (expected
   number of retained coefficients).
Output: Array $\Lambda = [\lambda_0, \ldots, \lambda_{N-1}]$ of rounding values for the nonzero coefficients (for zero
   coefficients, set to $\perp$) that minimizes the expected $L^2$ error under the constraints $\sum_i \frac{c_i}{\lambda_i} \leq B$
   and $0 < \frac{c_i}{\lambda_i} \leq 1$.
begin
1.  total := 0,   avail := $B$
2.  for i := 0 to $N - 1$ do
3.      sqrtk[$i$] = $\frac{|W_A[i]|}{2^{level(i)/2}}$                /* = $\sqrt{\frac{W_A[i]^2}{2^{level(i)}}}$ */
4.      total := total + sqrtk[$i$]
5.  endfor
6.  sort the indices $i$ in nonincreasing order of sqrtk[$i$] and place in an array $I$
7.  for $r$ := 0 to $N - 1$ do
8.      if (sqrtk[$I[r]$] $*$ avail $<$ total) then break          /* $\frac{c_i}{\lambda_i} < 1$ */
9.      else
10.         $\Lambda[I[r]]$ := $W_A[I[r]]$                         /* $\frac{c_i}{\lambda_i} \geq 1$, so set $\lambda_i := c_i$ */
11.         total := total $-$ sqrtk[$I[r]$],   avail := avail $-$ 1   /* reduce available space by 1 */
12.     endif
13. endfor
14. for $j$ := $r$ to $N - 1$ do        /* the subproblem remaining is solved using Equation 7 */
15.     if ($W_A[I[j]] \neq 0$) then
16.         $\Lambda[I[j]]$ := $\frac{W_A[I[j]] * \text{total}}{\text{sqrtk}[I[j]] * \text{avail}}$          /* guaranteed that $\frac{c_i}{\lambda_i} < 1$ */
17.     else $\Lambda[I[j]]$ := $\perp$
18. endfor
end
```

Fig. 2. The MinL2 Algorithm: Rounding to Minimize the Expected Mean-Squared Error.

This leads to our MinL2 algorithm, which we sketch here. We first compute the $\sqrt{k_i}$'s and their sum. We would like to apply Eq. (7) to set the rounding values, but this may result in one or more $y_i > 1$. Thus, we consider the indices i in nonincreasing order of $\sqrt{k_i}$. While $y_i \geq 1$, we set λ_i to c_i, so that $y_i = 1$, and then loop to recurse on the subproblem without i. Once we encounter a $y_i < 1$, we are guaranteed that all remaining y_i are less than 1 as well, and we can safely apply Eq. (7) to set the rounding values. Using convexity arguments, it can be shown that MinL2 produces the optimal solution to our optimization problem. The detailed pseudo-code for our MinL2 algorithm is depicted in Figure 2; the algorithm takes as input the coefficient array and the target space budget B, and outputs the rounding values that solve the optimization problem described above. Our MinL2 algorithm runs in linear time, plus the time for sorting (a total of $O(N \log N)$ time). It uses $O(N)$ total storage space. Using the rounding values produced by the algorithm results in unbiased approximate answers for individual values and for ranges, with the minimum expected mean-squared error over all individual values.

3.4 Rounding to Minimize the Maximum Relative Error

In the previous section, we described how to obtain unbiased approximate answers while minimizing an *overall* error metric. However, there was no attempt to minimize *individual* answer errors, and indeed there can be wide discrepancies in the accuracy of reconstructed values. (Recall from Section 3.1 that conventional wavelet synopses suffer from this same problem, among others.)

In this section, we present techniques for minimizing the maximum reconstruction error for individual data values and ranges. We focus on minimizing the *relative error*, and then describe our approach for *absolute error* in Section 3.5 that follows. Although minimizing absolute error is somewhat easier to achieve, we believe that minimizing relative error is more important to approximate query answering.[4] On the other hand, relative error is unduly dominated by small data values (e.g., for a data value = 3, returning 2 as the reconstructed value is a 50% relative error!), so various techniques have been studied for combining absolute and relative error (see, e.g., Haas and Swami [1992] and Vitter and Wang [1999]). In this article, we study an error metric that combines relative error with a *sanity bound* S. Our goal is to produce estimates $\hat{d}_i$ for each data value d_i such that, for a given sanity bound $\mathrm{S} > 0$, the ratio

$$\frac{|\hat{d}_i - d_i|}{\max\{|d_i|, \mathrm{S}\}}$$

is *small with high probability* (subject to the prescribed space limit for the synopsis).

Note that our probabilistic wavelet synopses guarantee that the expected value of $\hat{d}_i$ is d_i; thus, a simple application of Chebyshev's Inequality gives that, for any (small) constant $\epsilon > 1$,

$$\Pr\left(\frac{|\hat{d}_i - d_i|}{\max\{|d_i|, \mathrm{S}\}} \le \epsilon \cdot \textsc{nse}(\hat{d}_i)\right) \ge 1 - \frac{1}{\epsilon^2}\,, \tag{8}$$

where NSE denotes the *normalized standard error* for a reconstructed data value:

$$\textsc{nse}(\hat{d}_i) = \frac{\sqrt{\mathrm{Var}(\hat{d}_i)}}{\max\{|d_i|, \mathrm{S}\}}. \tag{9}$$

Therefore, minimizing NSE will minimize our relative error metric for a given level of confidence (see Eq. (8)).

Based on the earlier development, and letting $\mathtt{PATHS} = \{\mathtt{path}(d_i) : i = 1, \ldots, N\}$ (i.e., the set of all root-to-leaf paths in the error tree, where paths again ignore both data value nodes and nodes whose coefficient is zero), and applying Eq. 5, we can define our maximum NSE minimization problem as follows.

[Maximum Normalized Standard Error Minimization]. Find the rounding values λ_i that *minimize* the maximum NSE for each reconstructed

[4]For example, when (exact) answers can vary by orders of magnitude, it is often preferable to have each answer be within say 1% relative error than have each answer be within the same absolute error, because the same absolute error may correspond to orders of magnitude differences in relative error (say, .1% to 100%).

data value; that is,

$$\text{Minimize} \max_{\mathtt{path}(d_k)\, \in\, \mathtt{PATHS}} \frac{\sqrt{\sum_{i\, \in\, \mathtt{path}(d_k)} (\lambda_i - c_i) \cdot c_i}}{\max\{|d_k|, \mathtt{S}\}} \tag{10}$$

subject to the constraints $0 < \frac{c_i}{\lambda_i} \leq 1$ for all nonzero c_i and $E[|\mathtt{WS}_A|] = \sum_{i|c_i \neq 0} \frac{c_i}{\lambda_i} \leq B$. □

Our solution to the maximum NSE minimization problem relies on applying four key technical ideas, which we will describe throughout this section. The first is *Exploiting the Error-Tree Structure for Coefficients:* As explained above, the variance for the reconstruction of individual data values is computed by summing the contributions of independent random variables C_i along all root-to-leaf paths in the error-tree structure for Haar wavelet coefficients; our algorithm takes advantage of this hierarchical problem structure to efficiently explore the solution space using *dynamic programming*, as discussed next.

3.4.1 *Formulating a Dynamic-Programming Recurrence.* We would like to formulate a dynamic-programming recurrence for this problem. Accordingly, we first simplify Eq. (10) by squaring the main ($\text{NSE}(\hat{d}_k)$) term; this does not alter the optimality of any solution. As before, we let $y_i = \frac{c_i}{\lambda_i}$, where $y_i \in (0, 1]$ for a non-zero coefficient c_i. Then let $\text{VAR}(i, y_i) = (\lambda_i - c_i)c_i = \frac{1-y_i}{y_i} \cdot c_i^2$ denote the variance of C_i (the random variable associated with c_i) for the given y_i. Let T_j be the subtree of the error tree rooted at the node corresponding to coefficient c_j. Let $\mathtt{PATHS}_j$ denote the set of all root-to-leaf paths in T_j. Finally, let $M[j, B]$ denote the optimal (i.e., minimum) value of the maximum $\text{NSE}(\hat{d}_k)^2$ among all data values d_k in T_j assuming a space budget of B; that is,

$$M[j, B] = \min_{\substack{\text{all possible } y_i \in (0,1],\ i \in T_j | c_i \neq 0 \\ \text{such that } \sum_{i \in T_j | c_i \neq 0} y_i \leq B}} \left\{ \max_{\mathtt{path}(d_k) \in \mathtt{PATHS}_j} \sum_{i \in \mathtt{path}(d_k)} \frac{\text{VAR}(i, y_i)}{\max\left\{d_k^2, \mathtt{S}^2\right\}} \right\} \tag{11}$$

Consider the following recurrence for $M[j, B]$ (omitting the $j = 0$ case):

$$M[j, B] = \begin{cases} \min_{\substack{y_j \in (0, \min\{1, B\}]; \\ b_L \in [0, B - y_j]}} \left\{ \max \left\{ \begin{array}{l} \frac{\text{VAR}(j, y_j)}{\text{NORM}(2j)} + M[2j, b_L], \\ \frac{\text{VAR}(j, y_j)}{\text{NORM}(2j+1)} + M[2j+1, B - y_j - b_L] \end{array} \right\} \right\}, & \text{if } j < N, c_j \neq 0, \text{ and } B > 0 \\ \min_{b_L \in [0, B]} \left\{ \max\{M[2j, b_L], M[2j+1, B - b_L]\} \right\}, & \text{if } j < N \text{ and } c_j = 0 \\ 0 & \text{if } j \geq N \\ \infty & \text{otherwise} \end{cases} \tag{12}$$

where $\text{NORM}(i) = \max\{\min_{k \in T_i}\{d_k^2\}, \mathtt{S}^2\}$ is the normalization term for subtree T_i.

Note that the indices $2j$ and $2j+1$ in the above recurrence correspond to the left and right child (respectively) of node j in the error-tree structure (Figure 1). Intuitively, Eq. (12) states that, given a space budget of B for the subtree rooted at node j, the optimal solution for the y_i's and the corresponding cost (maximum NSE^2) needs to minimize the larger of the cost for paths via j's two child subtrees (including the root in all paths), where the cost for a path via a subtree is the sum of: (1) the variance penalty incurred at node j itself, assuming a setting of y_j, divided by the normalization term for that subtree, and (2) the optimal cost for the subtree, assuming the given space budget. This minimization, of course, is over all possible values of y_j and, given a setting of y_j, over all possible allotments of the remaining $B - y_j$ budget "units" amongst the two child subtrees of node j. Of course, if c_j is zero then none of our space budget needs to be allocated to node j, which results in the simpler recurrence in the second clause of Eq. (12). Finally, data nodes ($j \geq N$) cost no space and incur no cost, and the "otherwise" clause handles the case where we have a nonzero coefficient but zero budget ($c_j \neq 0$ and $B = 0$).

Eq. (12) is a significant simplification of Eq. (11), so why might it be correct? Consider a node $j \neq 0$ with coefficient $c_j \neq 0$. Starting with Eq. (11) and separating the root j from the rest of the path, we can observe that

$$M[j, B] = \min_{\substack{y_j \in (0, \min\{1,B\}];\\ b_L \in [0, B-y_j]}} \left\{ \max \left\{ \begin{array}{l} \displaystyle\max_{\text{path}(d_L)\in\text{PATHS}_{2j}} \left\{ \frac{\text{VAR}(j,y_j)}{\max\{d_L^2, \text{S}^2\}} + \sum_{i\in\text{path}(d_L)} \frac{\text{VAR}(i, y_i(b_L))}{\max\{d_L^2,\text{S}^2\}} \right\}, \\ \displaystyle\max_{\text{path}(d_R)\in\text{PATHS}_{2j+1}} \left\{ \frac{\text{VAR}(j,y_j)}{\max\{d_R^2,\text{S}^2\}} + \right. \\ \qquad\qquad\qquad \displaystyle\left. \sum_{i\in\text{path}(d_R)} \frac{\text{VAR}(i, y_i(B-y_j-b_L))}{\max\{d_R^2,\text{S}^2\}} \right\} \end{array} \right\} \right\} \tag{13}$$

where the $y_i(b)$'s are the optimal choice for y_i's given space b. When $c_j = 0$, this simplifies to

$$M[j, B] = \min_{b_L \in [0,B]} \left\{ \max \left\{ \begin{array}{l} \displaystyle\max_{\text{path}(d_L)\in\text{PATHS}_{2j}} \left\{ \sum\nolimits_{i\in\text{path}(d_L)} \frac{\text{VAR}(i, y_i(b_L))}{\max\{d_L^2,\text{S}^2\}} \right\}, \\ \displaystyle\max_{\text{path}(d_R)\in\text{PATHS}_{2j+1}} \left\{ \sum\nolimits_{i\in\text{path}(d_R)} \frac{\text{VAR}(i, y_i(B-b_L))}{\max\{d_R^2,\text{S}^2\}} \right\} \end{array} \right\} \right\} \tag{14}$$

To proceed from here, we need to apply a second key technical idea: *Exploiting Properties of Optimal Solutions.* Consider the following claim:

CLAIM 3.3. *Consider a subtree T_j and a space budget $B > 0$. Then, in an optimal setting of the y_i's, a minimum data value $d_{\min}$ in T_j has the maximum cost, that is,* NSE$(\hat{d}_{\min})^2 = M[j, B]$.

Assuming for now that Claim 3.3 is correct, we have, for all $d_L \in T_{2j}$:

$$\frac{\text{VAR}(j, y_j)}{\max\{d_{\min}^2, \text{S}^2\}} = \frac{\text{VAR}(j, y_j)}{\text{NORM}(2j)} \geq \frac{\text{VAR}(j, y_j)}{\max\{d_L^2, \text{S}^2\}},$$

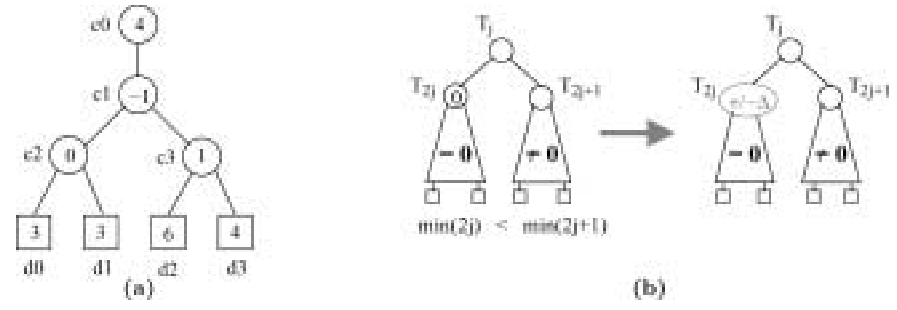


Fig. 3. (a) A problem error tree for Claim 3.3. (b) An illustration of our Perturbation Rule.

and

$$\sum_{i \in \mathtt{path}(d_{\min})} \frac{\textsc{var}(i,\ y_i(b_L))}{\max\{d_{\min}^2, \mathrm{S}^2\}} \ = \ M[2j, b_L] \ \geq \ \sum_{i \in \mathtt{path}(d_L)} \frac{\textsc{var}(i,\ y_i(b_L))}{\max\{d_L^2, \mathrm{S}^2\}}.$$

Thus, the maximum cost path via the left subtree is $\mathtt{path}(d_{\min})$, and its cost is $\frac{\mathrm{VAR}(j, y_j)}{\mathrm{NORM}(2j)} + M[2j, b_L]$. By a similar argument for the right subtree, it follows that Eq. (12) is correct, assuming Claim 3.3 is correct.

Unfortunately, there are corner cases which make Claim 3.3 false! Consider, for example, the error tree depicted in Figure 3(a). For subtree T_1, given a space budget $B = 1$ and a sanity bound $\mathrm{S} = 1$, the optimal setting of the y_i's is $y_1 = \frac{1}{2}$, $y_2 = 0$, and $y_3 = \frac{1}{2}$. (We leave it to the reader to verify that this setting indeed minimizes the cost for T_1 for this B and S.) Note that $\textsc{var}(1, \frac{1}{2}) = \frac{1-\frac{1}{2}}{\frac{1}{2}}(-1)^2 = 1$ and $\textsc{var}(3, \frac{1}{2}) = \frac{1-\frac{1}{2}}{\frac{1}{2}}(1)^2 = 1$. For the minimum data values $d0$ and $d1$, we have $\textsc{nse}(\hat{d0})^2 = \textsc{nse}(\hat{d1})^2 = \frac{\mathrm{VAR}(1, \frac{1}{2})}{\max\{3^2, 1^2\}} = \frac{1}{9}$ for T_1. On the other hand, $\textsc{nse}(\hat{d3})^2 = \frac{\mathrm{VAR}(1, \frac{1}{2}) + \mathrm{VAR}(3, \frac{1}{2})}{\max\{4^2, 1^2\}} = \frac{1}{8}$ for T_1. So, $M[1, 1] = \frac{1}{8} > \textsc{nse}(\hat{d}_{\min})$, contradicting Claim 3.3.

Figure 3(a) provides a simple example of a family of error trees that violate Claim 3.3. The problem arises because the subtree containing $d_{\min}$ (in this case T_2) has all 0 coefficients, but its sibling subtree (in this case T_3) has some nonzero coefficients. In the worst case, this problem compounds itself up the tree, destroying any hope of an effective dynamic-programming formulation. Thus, we need to apply a third key technical idea: *Perturbing Coefficients to Avoid Harmful Corner Cases*. We can ensure these corner case do not occur by "perturbing" certain zero coefficients by a very small perturbation amount Δ, making them either $+\Delta$ or $-\Delta$ with equal probability (to ensure no reconstruction bias is introduced):

[Perturbation Rule]. For each subtree T_j such that (1) one of its child subtrees, say T_{2j}, has all zero coefficients, (2) its other child subtree, T_{2j+1}, has at least one nonzero coefficient, and (3) the minimum data value in T_{2j} is less than the minimum data value in T_{2j+1}, we perturb c_{2j}.

Our perturbation rule is pictorially depicted in Figure 3(b). Note that we can determine which coefficients need to be perturbed in linear time (actually, $O(N_z)$ time, where N_z is the number of nonzero cells): Simply compute bottom-up the minimum data value and the number of nonzero coefficients in each subtree, and then perform the test for each subtree. Moreover, each data value has at most one perturbed coefficient on its path, so the effect of this perturbation is minimal. We determined experimentally that a good choice for Δ is $\min\{0.01, \frac{\mathsf{S}}{100}\}$. This leads to the following lemma, which is a corrected version of Claim 3.3:

LEMMA 3.4. *Consider an error tree T after applying the Perturbation Rule. For each subtree T_j of T, $0 \le j < N$, and all $B > 0$, and considering $M[j, B]$ as defined in Eqs.* (13) *and* (14), *we have:*

$$M[j, B] = \sum_{i \in \mathtt{path}(d_{\min}) \in \mathtt{PATHS}_j} \frac{\textsc{var}(i, y_i(B))}{\max\{d_{\min}^2, \mathsf{S}^2\}}, \tag{15}$$

where $d_{\min}$ is a minimum data value in T_j and the $y_i(B)$'s are the optimal choice for y_i's given space B.

PROOF. Note first the following two observations:

O1. For $0 \le j < N$, when $B > 0$, $M[j, B]$ is finite. That is, the optimal allocation allots positive space to each nonzero coefficient, so that its variance is finite.

O2. For any subtree T_j with $k \ge 0$ nonzero coefficients, $M[j, b]$ is a positive, continuous, decreasing function of b over the range $0 < b < k$, and $M[j, b] = 0$ for $b \ge k$. That is, allotting more space to T_j decreases its cost until we have sufficient space to store all its nonzero coefficients, at which case the cost is zero.

In the argument that follows, we restrict our attention to j and B such that $1 \le j < N$ and $B > 0$. Extensions to handle $j = 0$ are straightforward.

The proof is by induction on the height h of the subtree. The base case $h = 1$ is a subtree T_j whose two children are data nodes, $d_{\min}$ and d, with $d_{\min} \le d$. Both $\mathtt{PATHS}_{2j}$ and $\mathtt{PATHS}_{2j+1}$ are empty, so by Eqs. (13) and (14),

$$M[j, B] = \min_{y_j \in (0, \min\{1, B\}]} \max\left\{ \frac{\textsc{var}(j, y_j)}{\max\{d_{\min}^2, \mathsf{S}^2\}}, \frac{\textsc{var}(j, y_j)}{\max\{d^2, \mathsf{S}^2\}} \right\} = \frac{\textsc{var}(j, \min\{1, B\}]}{\max\{d_{\min}^2, \mathsf{S}^2\}}.$$

If $c_j = 0$, then $M[j, B] = 0$ and $\mathtt{path}(d_{\min})$ is empty, so Eq. (15) holds. Otherwise, $c_j \ne 0$ and $\mathtt{path}(d_{\min}) = \{j\}$, so Eq. (15) holds. This proves the base case.

Assume the lemma holds for all subtrees of height $h \ge 1$. We will show it holds for any subtree T_j of height $h + 1$. Let d_L^* (d_R^*) be a minimum data value in T_{2j} (T_{2j+1}, respectively). Assume, without loss of generality, that $d_L^* \le d_R^*$. Let k_L (k_R) be the number of nonzero coefficients in T_{2j} (T_{2j+1}, respectively). Based on observation O1, we say an allotment is a *candidate* allotment if it assigns positive space to each nonzero coefficient.

Consider first the case when $k_L > 0$. By induction, for all allotments $b_L > 0$,

$$M[2j, b_L] = \sum_{i \in \mathtt{path}(d_L^*) \in \mathtt{PATHS}_{2j}} \frac{\textsc{var}(i, y_i(b_L))}{\max\{(d_L^*)^2, \mathsf{S}^2\}} \ge \sum_{i \in \mathtt{path}(d_L) \in \mathtt{PATHS}_{2j}} \frac{\textsc{var}(i, y_i(b_L))}{\max\{d_L^2, \mathsf{S}^2\}}$$

for all $d_L \in T_{2j}$ (since $M[2j, b_L]$ is the maximum). Moreover, for all candidate allotments y_j and all $d_L \in T_{2j}$, $\frac{\mathrm{VAR}(j, y_j)}{\max\{(d_L^*)^2, \mathsf{S}^2\}} \ge \frac{\mathrm{VAR}(j, y_j)}{\max\{d_L^2, \mathsf{S}^2\}}$, because $d_L^* \le d_L$. Thus, for all candidate allotments y_j,

$$\max_{\mathtt{path}(d_L)\in\mathtt{PATHS}_{2j}} \left\{ \frac{\textsc{var}(j,\, y_j)}{\max\{d_L^2, \mathrm{S}^2\}} + \sum_{i\in\mathtt{path}(d_L)} \frac{\textsc{var}(i,\, y_i(b_L))}{\max\{d_L^2, \mathrm{S}^2\}} \right\}$$
$$= \frac{\textsc{var}(j,\, y_j)}{\max\{(d_L^*)^2, \mathrm{S}^2\}} + M[2j, b_L].$$

Applying a similar argument to T_{2j+1}, we see that $M[j, B]$ minimizes the maximum of

$$\frac{\textsc{var}(j,\, y_j)}{\max\{(d_L^*)^2, \mathrm{S}^2\}} + M[2j, b_L] \tag{16}$$

and

$$\frac{\textsc{var}(j,\, y_j)}{\max\{(d_R^*)^2, \mathrm{S}^2\}} + M[2j+1, (B - y_j) - b_L]. \tag{17}$$

We claim that for any candidate allotment with a given y_j, there is an optimal choice b_L^* for b_L that minimizes the above maximum such that (16) is greater than or equal to (17). To see this, suppose instead that (17) is greater than (16) under the optimal allocation b_L^*. Then, because $d_L^* \le d_R^*$, we would have that $M[2j+1, (B-y_j) - b_L^*] > M[2j, b_L^*] \ge 0$. Because we consider only candidate allotments, and applying observation O2, we would have that $\textsc{var}(j,\, y_j)$ is finite, $0 < (B - y_j) - b_L^* < k_R$, and $b_L^* > 0$ (recall that $k_L > 0$). It follows, based again on observation O2, that we could shift some of the allotment from T_{2j} to T_{2j+1} and improve $M[j, B]$. Specifically, there would exist an allotment $\epsilon > 0$ such that $M[2j+1, (B-y_j) - b_L^* + \epsilon] < M[2j+1, (B-y_j) - b_L^*]$ and

$$\frac{\textsc{var}(j,\, y_j)}{\max\{(d_L^*)^2, \mathrm{S}^2\}} + M[2j, b_L^* - \epsilon] \le \frac{\textsc{var}(j,\, y_j)}{\max\{(d_R^*)^2, \mathrm{S}^2\}} + M[2j+1, (B-y_j) - b_L^* + \epsilon]$$

This contradicts b_L^* being the optimal allocation. Therefore, we conclude that (16) is greater than or equal to (17), and hence Eq. (15) holds for T_j, where $d_{\min} = d_L^*$.

Now consider the case when $k_L = 0$. If k_R also equals zero, the proof that Eq. (15) holds follows along the lines argued in the base case. So, consider the case when $k_R > 0$. Suppose $d_L^* < d_R^*$.[5] Then, the perturbation rule would apply to T_j, resulting in $c_{2j} \ne 0$, contradicting $k_L = 0$. Thus, $d_L^* = d_R^*$. In this case, the proof that Eq. (15) holds mimics the case when $k_L > 0$, with the roles of T_{2j} and T_{2j+1} reversed.

Therefore, in all cases, Eq. (15) holds for T_j. The lemma follows by induction. □

Given Lemma 3.4, it is now straightforward to prove the main theorem of this section, which shows that the dynamic-programming recurrence in Eq. (12)

[5]Note that, because $k_L = 0$, the above approach of shifting space from T_{2j} to T_{2j+1} will not help: T_{2j} may have no space allotted and hence no space to give, and even if it did have allotted space, reducing its space will not increase (16). In general, without the perturbation rule, the lemma would fail for this case (recall Figure 3(a)).

can be used to solve the Maximum Normalized Standard Error Minimization problem.

THEOREM 3.5. *Consider an error tree T after applying the Perturbation Rule. For all j, $0 \leq j < N$, and all $B \geq 0$, the dynamic-programming formulation in Eq.* (12) *correctly computes the optimal $M[j, B]$, as defined in Eq.* (11). *Hence, computing $M[0, B]$ using Eq.* (12), *while saving the optimal choices for y_j and b_L at each step, yields the optimal y_i's (and hence λ_i's) that solve the Maximum Normalized Standard Error Minimization problem.*

PROOF. We argued earlier that Eqs. (13) and (14) are equivalent to Eq. (11). It follows from Lemma 3.4 and the definition of NORM that Eq. (12) is equivalent to Eqs. (13) and (14). (The various corner cases, such as when $B = 0$, can be readily handled.) □

The problem with the recurrence in Eq. (12) is that the y_j and b_L each range over a continuous interval, making it infeasible to use. Thus, rather than insisting on an exact solution, we instead propose an efficient approximation algorithm that produces near-optimal solutions to our maximum NSE minimization problem.

3.4.2 *An Efficient Approximation Algorithm for Minimizing the Relative Error.* The fourth and final key technical idea applied in our dynamic-programming algorithm is to *Quantize the Solution Space:* Instead of allowing the y_i variables to vary continuously over (0, 1], we assume that these variables take values from a discrete set of q choices corresponding to integer multiples of $1/\mathsf{q}$, where q is an input integer parameter to our algorithm; that is, we modify the constraint $y_i \in (0, 1]$ to $y_i \in \{\frac{1}{\mathsf{q}}, \frac{2}{\mathsf{q}}, \ldots, 1\}$, and the constraint $b_L \in [0, B]$ to $b_L \in \{0, \frac{1}{\mathsf{q}}, \ldots, B\}$. Obviously, our approximate solution converges to the optimal solution for the maximum NSE minimization problem as q becomes larger. On the other hand, as we show below, larger values for q also imply higher running-time and memory requirements for our algorithm. Thus, the input "quantizing" parameter q provides a convenient "knob" for tuning the trade-off between resource requirements and solution quality for our approximate dynamic-programming algorithm.

The pseudo-code for our algorithm (termed MinRelVar) is based on the above quantization of the recurrence in Eq. (12), and is depicted in Figure 4. The initial invocation of this recursive algorithm is done with `root` $= 0$ and B equal to the total (expected) number of coefficients to be retained in the wavelet synopsis.[6]

3.4.3 *Time/Space Complexity and the Quantizing Parameter* q. Given a node (coefficient) j in the error tree and a budget B_j, MinRelVar computes the optimal value by examining q possible space allotments for j (Step 8) and, for each of these, a maximum of $O(\mathsf{q}B_j)$ ways of distributing the remaining budget among the two children of node j (Step 16). (Once an optimal value is computed

[6]Note that node 0, corresponding to the overall average, should be handled as a special case, since it only has one child in the error tree (Figure 1); the modifications required are straightforward and omitted from the description for clarity.

```
procedure MinRelVar( $W_A$ , $B$ , q, root )
Input: Array $W_A = [c_0, \ldots, c_{N-1}]$ of $N$ Haar wavelet coefficients, space budget $B$ (expected
  number of retained coefficients), quantizing parameter $q \geq 1$, error-subtree root-node index
  root.
Output: Value of $M$[root, $B$] according to the quantized version of Equation 12 (denoted
  $M$[root, $B$].value), together with the corresponding optimal space allotments for the root
  (denoted $M$[root, $B$].yValue) and for the left child subtree (denoted $M$[root, $B$].leftAllot).
Note: We assume that prior to the first invocation of this procedure, the perturbation rule has
  been applied to the coefficients, using a given perturbation value parameter $\Delta$. Also, both
  NORM($i$), using a given sanity bound parameter S, and $NZ[i]$, the number of nonzero coeffi-
  cients in the subtree rooted at node $i$, have been computed for all $i$. Finally, $M[i, B]$.computed
  is initialized to false, for all $i$ and all $B = 0, \frac{1}{q}, \ldots, B^*$, where $B^*$ is the root's space budget.
  Note that the pseudo-code does not explicitly handle the special case where root is the overall
  root of the error-tree.
begin
1.  if (root $> N - 1$ or $NZ$[root] $\leq B$) then
2.      return 0                  /* nothing more to round or enough space to retain all coefficients */
3.  if ($NZ$[root] $> B * q$) then
4.      return $\infty$                 /* not enough space even for minimal allocation to each coefficient */
5.  if ($M$[root, $B$].computed = true) then
6.      return $M$[root, $B$].value    /* optimal value already in the dynamic-programming array */
7.  $M$[root, $B$].value := $\infty$
8.  for $l$ := 1 to q do
9.      if ($c_{\texttt{root}}$ = 0) then
10.         rootLeft := rootRight := rootSpace := 0
11.     else
12.         rootLeft := $(q - l) * c^2_{\texttt{root}}/(l * \text{NORM}(2 * j))$
13.         rootRight := $(q - l) * c^2_{\texttt{root}}/(l * \text{NORM}(2 * j + 1))$
14.         rootSpace := $l/q$
15.     endif
16.     for $b$ := 0 to $B-$rootSpace step 1/q do
17.         left := MinRelVar( $W_A$, $b$, q, $2 *$ root )
18.         right := MinRelVar( $W_A$, $B-$rootSpace$-b$, q, $2 *$ root $+ 1$ )
19.         if ( max{ rootLeft+left , rootRight+right } $<$ $M$[root, $B$].value ) then
20.             $M$[root, $B$].value := max{ rootLeft+left , rootRight+right }
21.             $M$[root, $B$].yValue := rootSpace
22.             $M$[root, $B$].leftAllot := b
23.         endif
24.     endfor
25.     if ($c_{\texttt{root}}$ = 0) then break        /* no need to iterate over multiple $l$ */
26. endfor
27. $M$[root, $B$].computed := true
28. return $M$[root, $B$].value
end
```

Fig. 4. The MinRelVar Algorithm: Rounding to Minimize Maximum Normalized Standard Error.

it is recorded in the dynamic-programming array for future reference.) To simplify the exposition, our pseudo-code in Figure 4 gives a simple $O(qB_j)$-time linear-search procedure for finding the optimal distribution of space budget among the two children of node j, that is, the distribution that minimizes the maximum error for the two child subtrees (Steps 16–24). Note, however, that this search can actually be performed using a more efficient $O(\log(qB_j))$-time *binary-search* procedure. The key observation here is that, for any subtree (rooted at `root`), $M[\texttt{root}, B]$ is a decreasing function of the budget B (until $B \geq NZ[\texttt{root}]$, at which point it stays constant at 0). Thus, the optimal distribution point (i.e., the best leftAllot value in Step 22) for a given rootSpace can be found by binary search, looking for the allotment where the error values for the two child subtrees are equal or for the adjacent pair of cross-over allotments. This implies that the search at node j for a given budget B_j can be performed using only $O(\mathrm{q} \log(\mathrm{q}B_j))$ accesses to the dynamic-programming array M. There are N nodes and $\mathrm{q}B$ choices for B_j, so the overall running-time complexity of algorithm MinRelVar is $O(N\mathrm{q}^2 B \log(\mathrm{q}B))$. Moreover, typically the running time will be faster due to the considerable pruning during the search. With respect to the space requirements of our algorithm, note that, even though the size of the full dynamic-programming array M is $O(N\mathrm{q}B)$, MinRelVar does not require the entire array to be memory resident. In fact, it is easy to verify that, at any given point during the execution of MinRelVar, there will be at most one active "line" (of size $O(\mathrm{q}B)$) of array M per level of the error tree. This is because the results for all descendants of a node j can be swapped out once the results for j have been computed. Thus, the memory resident working set size is only $O(\mathrm{q}B \log N)$.[7]

3.4.4 *An Optimization.* Recall that for nonzero coefficients, we select a $y_i \in \{\frac{1}{\mathrm{q}}, \frac{2}{\mathrm{q}}, \ldots, 1\}$. Because it is often useful to permit very small y_i's for unimportant coefficients, this forces a larger choice for q, in order to make the smallest value, $\frac{1}{\mathrm{q}}$, be sufficiently small. Instead, we propose the following simple optimization: allow $y_i = 0$ even for nonzero coefficients. The problem, of course, is that then the variance, $\textsc{var}(i, y_i) = \frac{1-y_i}{y_i} \cdot c_i^2$, for this coefficient is infinite, and hence this choice for y_i will never be selected as the optimal choice by the MinRelVar algorithm. To get around this problem, we observe that because the coefficient is always rounded down when $y_i = 0$, its contribution to the squared error of any data value in its subtree is c_i^2, not infinite. Thus, we can set $\textsc{var}(i, 0) = c_i^2$ in our dynamic-programming algorithm. The downside of this optimization is that it introduces a small bias in the reconstruction. On balance, however, it leads to a faster algorithm (because a larger q can be used) and highly accurate answers (see Section 4).

3.5 Rounding to Minimize the Maximum Absolute Error

In this section, we present an efficient approximation algorithm for minimizing the absolute error, based on dynamic programming. Consider a data value d_i and its estimate $\hat{d}_i$ based on our probabilistic wavelet synopses. As in Section 3.4, applying Chebyshev's Inequality we have that, for any (small)

[7]If we desire to limit the total space (not just the working set size) to $O(N + \mathrm{q}B \log^2 N)$, then a bottom-up dynamic-programming algorithm achieves this bound, because we can *discard* the results for a node's two children once its results are computed. The only information we would later need from the children is the yValue's and leftAllot's on the optimal path to a leaf. Retaining these paths is the source of the extra $\log N$ in the space bound. The drawback of this approach is that we do *no pruning*, so that the running time is $\Theta(N\mathrm{q}^2 B \log(\mathrm{q}B))$.

constant $\epsilon > 1$,

$$\Pr\left(|\hat{d}_i - d_i| \le \epsilon \cdot \sqrt{\mathrm{Var}(\hat{d}_i)}\right) \ge 1 - \frac{1}{\epsilon^2}.$$

Thus, by Eq. (5), it is easy to see that choosing rounding values to minimize the maximum absolute error is essentially equivalent to minimizing the *maximum total variance across all root-to-leaf paths* in the error tree of the Haar wavelet decomposition.

[Maximum Reconstruction Variance Minimization]. Find the rounding values λ_i that *minimize* the maximum value of the variance for each reconstructed data value; that is,

$$\text{Minimize} \max_{p \in \mathtt{PATHS}} \sum_{i \in p} \mathrm{Var}(C_i) = \max_{p \in \mathtt{PATHS}} \sum_{i \in p} (\lambda_i - c_i) \cdot c_i$$

subject to the constraints $0 < \frac{c_i}{\lambda_i} \le 1$ for all non-zero c_i and $E[|\mathtt{WS}_A|] = \sum_{i | c_i \neq 0} \frac{c_i}{\lambda_i} \le B$. $\square$

As with the maximum NSE minimization problem formulated in Section 3.4, the above non-linear optimization problem is significantly more difficult to solve than our earlier expected L^2 error minimization. Fortunately, we can once again apply the key dynamic-programming and quantization ideas introduced in Section 3.4 to design an efficient approximation algorithm for our maximum variance minimization problem. Specifically, let, once again, $y_i = \frac{c_i}{\lambda_i}$ denote the "success" probability of random variable C_i, and let $M[j, B]$ denote the optimal (i.e., minimum) value of the maximum variance among all paths in $\mathtt{PATHS}_j$ assuming a space budget of B and a quantization parameter q; that is,

$$M[j, B] = \min_{\substack{\text{all possible } y_i \in \{\frac{1}{\mathrm{q}}, \ldots, 1\},\, i \in T_j | c_i \neq 0 \\ \text{such that } \sum_{i \in T_j | c_i \neq 0} y_i \le B}} \left\{ \max_{p \in \mathtt{PATHS}_j} \sum_{i \in p} \frac{1 - y_i}{y_i} \cdot c_i^2 \right\}.$$

Then, $M[j, B]$ can be computed using the following dynamic-programming recurrence:

$$M[j, B] = \begin{cases} \min\limits_{\substack{l = 1, \ldots, \mathrm{q}; \\ b_L = 0, \frac{1}{\mathrm{q}}, \ldots, B - \frac{l}{\mathrm{q}}}} \left\{ \frac{\mathrm{q} - l}{l} \cdot c_j^2 + \max\{M[2j, b_L], M[2j+1, B - \frac{l}{\mathrm{q}} - b_L]\} \right\}, & \\ & \text{if } c_j \neq 0 \\ \min\limits_{b_L = 0, \frac{1}{\mathrm{q}}, \ldots, B} \left\{ \max\{M[2j, b_L], M[2j+1, B - b_L]\} \right\} & \text{if } c_j = 0 \end{cases} \tag{18}$$

Intuitively, Eq. (18) states that, given a space budget of B for the subtree rooted at node j, the optimal solution for the y_i's and the corresponding maximum path variance subtree needs to minimize the sum of two quantities: (1) the variance penalty incurred at node j itself, assuming a setting of $y_j = \frac{l}{\mathrm{q}}$ (the first term in the sum of Eq. (18), where $\frac{1 - y_j}{y_j} = \frac{\mathrm{q} - l}{l}$), and (2) the maximum of the optimal path variances for the two subtrees rooted at the two children of node j (the second summand in Eq. (18)); this minimization is over all the q possible values of y_j and, given a setting of $y_j = \frac{l}{\mathrm{q}}$, over all possible allotments of the remaining $B - \frac{l}{\mathrm{q}}$ budget "units" amongst the two child subtrees of node j. Of course, if c_j is zero then none of our space budget needs to be allocated to node j, which results in the simpler recurrence in the second clause of Eq. (18).

Based on the above recurrence, we can formulate a dynamic-programming algorithm (similar to our MinRelVar algorithm) to efficiently produce an approximate solution to the maximum variance minimization problem. Of course, note that, since we are no longer normalizing the path summations by the corresponding data values, our algorithm can directly apply the dynamic-programming recurrence in Eq. (18) without requiring any zero-coefficient perturbations. As in Section 3.4, we can show that our maximum variance minimization algorithm runs in $O(N\mathrm{q}^2 B \log(\mathrm{q}B))$ time and uses $O(N + \mathrm{q}B\log^2 N)$ space.

3.6 Low-Bias Probabilistic Wavelet Synopses

A key feature of our probabilistic wavelet synopses is their use of randomized rounding, in order to achieve unbiased reconstruction of individual values and range query results. In this section, we propose an alternative scheme that does not perform randomized rounding. Instead, each coefficient is either retained or discarded, according to the probabilities y_i, where as before the y_i's are selected to minimize a desired error metric.

The high-level approach is the same as before, except that now there is a random variable, C_i, associated with each coefficient that is c_i with probability y_i, and 0 with probability $1 - y_i$. Clearly, C_i is no longer an unbiased estimator for c_i, so this scheme introduces bias into the reconstruction of data values. To combat this, we select y_i's to minimize the maximum normalized bias for a reconstructed data value, where the *normalized bias* for a data value d_k is

$$\frac{\sum_{i\,\in\,\mathtt{path}(d_k)} |c_i| \cdot (1 - y_i)}{\max\{|d_k|, \mathrm{S}\}}.$$

To see why the $|c_i| \cdot (1 - y_i)$ term above makes sense, observe that $E[C_i] = c_i \cdot y_i$. It follows that each C_i contributes either plus or minus $c_i(1 - y_i)$ to the expected reconstruction bias of all data values in its subtree. Thus, $|c_i| \cdot (1 - y_i)$ upper bounds the expected contribution of this coefficient to the reconstruction bias.

We define our maximum normalized bias minimization problem as follows.

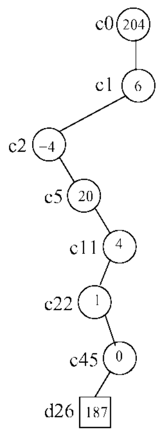
[Maximum Normalized Bias Minimization]. Find the y_i's that *minimize* the maximum normalized bias for each reconstructed data value; that is,

$$\text{Minimize} \quad \max_{\mathtt{path}(d_k)\,\in\,\mathtt{PATHS}} \frac{\sum_{i\,\in\,\mathtt{path}(d_k)} |c_i| \cdot (1 - y_i)}{\max\{|d_k|, \mathrm{S}\}} \tag{19}$$

subject to the constraints $0 \le y_i \le 1$ for all i, and $E[|\mathtt{WS}_A|] = \sum_i y_i \le B$. □

We can readily adapt the combinatorial solution of Section 3.4 to solve this minimization problem, and then quantize the solution space to obtain a tunable approximation algorithm. The dynamic programming formulation for $M[j, B]$

i	0	1	2	5	11	22	45
c_i	204	-6	-4	20	4	-1	0
y_i	1	$\frac{2}{3}$	$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{6}$	–
λ_i	204	-9	-8	20	8	-6	$\perp$
coins	S	S	F	S	S	F	–
WS	204	-9	–	20	8	–	–



Overall Algorithm:

Step 1. Select y_i's using the desired optimization algorithm (e.g., MinRelVar). Defines each $\lambda_i = \frac{c_i}{y_i}$ when using rounding (shown), or $\lambda_i = c_i$ when not using rounding.

Step 2. Flip coins, which defines the synopsis WS.

Steps 3+. Answer queries from WS by applying properties (P1) and (P2). E.g., $\hat{d}_{26} = 204 - 9 - 20 + 8 = 183$. (NB: actual value for $d_{26} = 187$).

Fig. 5. Summary of the approach, with an example (error bounds not shown).

(omitting the $j = 0$ case) is:

$$M[j,B] = \begin{cases} \displaystyle\min_{\substack{y_j \in \{\frac{1}{q},\frac{2}{q},\ldots,\min\{1,B\}\};\\ b_L \in \{0,\frac{1}{q},\ldots,B-y_j\}}} \left\{ \max\left\{ \begin{array}{l} \frac{(1-y_j)\cdot|c_j|}{\text{NORM}^*(2j)} + M[2j, b_L], \\ \frac{(1-y_j)\cdot|c_j|}{\text{NORM}^*(2j+1)} + M[2j+1, B-y_j-b_L] \end{array} \right\} \right\}, & \\ & \text{if } c_j \neq 0 \\ \min_{b_L \in \{0,\frac{1}{q},\ldots,B\}} \left\{ \max\{M[2j, b_L], M[2j+1, B-b_L]\} \right\} & \text{if } c_j = 0 \end{cases}$$

where $\text{NORM}^*(i) = \max\{\min_{k\in T_i}\{d_k\}, \mathrm{S}\}$ is the normalization term for subtree T_i. We will refer to the algorithm based on this formulation as the MinRelBias algorithm.

Intuitively, this approach has more in common with traditional wavelet synopses, in that we either retain or discard a coefficient as is. However, due to the randomization and our choice of optimization metric, we still avoid the three pitfalls of traditional wavelet synopses outlined in Section 1. In fact, as shown in Section 4, the MinRelBias algorithm produces probabilistic wavelet synopses that yield significantly more accurate answers than traditional wavelet synopses, and often outperform our other approaches.

3.7 Summary of the Approach

Figure 5 summarizes the steps for constructing and using our probabilistic wavelet synopses. For brevity, we show example values for c_i, λ_i, etc. only for a single path in an example error tree. In Step 1, we select the y_i using any of the optimization algorithms (e.g., MinL2, MinRelVar or MinRelBias). This defines the $\lambda_i = \frac{c_i}{y_i}$ when using rounding, or $\lambda_i = c_i$ when not using rounding. In Step 2, we randomly either retain or discard each λ_i, according to the probability y_i. An example outcome is shown (labeled "coins"). This results in the probabilistic wavelet synopsis $\{(0, 204), (1, -9), (5, 20), (11, 8), \ldots\}$. In Steps 3+, the synopsis is used to answer queries. For point queries, property (P1) is used. For range sums or averages, property (P2) is used. More generally, we can apply any of the previous techniques of Chakrabarti et al. [2000] and Vitter and Wang [1999] for answering queries from wavelet synopses (treating our probabilistic wavelet synopsis as a conventional wavelet synopsis).

In order to provide error guarantees for the approximation of individual data values, we can compute the maximum relative error in our approximate-value reconstruction and return it to the user as a *guaranteed relative-error bound* for any data value in the underlying domain. Similarly, instead of the worst-case relative error, we can report the 10th-percentile (or, some other quantile) of the relative error over all values as a (tighter) upper bound on the approximation for 90% of the values. Of course, conventional wavelet-thresholding techniques can also return similar error bounds but, since our probabilistic wavelet synopses (e.g., built using the MinRelVar and MinRelBias techniques) are specifically designed to optimize relative-error metrics, we expect our error guarantees to be much tighter and more informative for users. (Our experimental results in Section 4 clearly substantiate our claims.) Providing error guarantees for range queries (e.g., range-SUM aggregates) based on our synopses, is slightly more complicated. It is easy to see that for any range containing values that are all greater than our specified sanity bound S, our maximum value-reconstruction relative error guarantee carries over directly to the corresponding range aggregate. For arbitrary ranges, one possibility is to maintain the maximum (or, e.g., 10th-percentile) relative error over all possible $O(N^2)$ range aggregates in the underlying domain and return that as an error guarantee to the user. Once again, our experimental results (Section 4) show that such range-aggregate guarantees based on our synopses are much tighter and more informative than those for conventional wavelets. Obtaining more effective error guarantees for range aggregates may be possible by allocating some additional space in our synopses for this purpose; this could be an interesting direction for future work in this area.

4. EXPERIMENTAL STUDY

In this section, we present the results of an empirical study we conducted using the techniques developed in this paper for building probabilistic wavelet synopses. The objective of this study is to verify the effectiveness of our probabilistic synopsis techniques in reducing the errors in (a) *data-value reconstruction*, and (b) *approximate range-aggregate query answers*, compared to conventional, deterministic thresholding based on normalized coefficient values. To this end, we have experimented with different synthetic and real-world data sets. The major findings of our study can be summarized as follows:

—*More Consistent, Low-Error Data Reconstruction and Range-Aggregate Approximation.* By exploiting our randomized thresholding strategies, probabilistic wavelet synopses can enable a more consistent, lower-error approximation of both individual data values and range-aggregate answers; the end result is a significantly smaller mean relative error in data reconstruction

Table VI. Errors with Conventional and Probabilistic Wavelet Synopses

Original data values	127	71	87	31	59	3	43	99
	100	42	0	58	30	88	72	130
Deterministic answers	65	65	65	65	65	65	65	65
	100	42	0	58	30	88	72	130
MinRelVar answers	79	79	79	79	59	3	71	71
	71	71	0	58	31.7	128.3	80	80
MinRelBias answers	79	79	79	79	59	3	71	71
	71	71	0	58	51	109	80	80

and range-aggregate approximation across the entire underlying data domain.

—*Improved Quality Guarantees for Individual Data Values and Range-Aggregate Query Answers.* Besides reducing the overall relative error, our probabilistic wavelet synopses can also significantly reduce the maximum (and other percentiles) of relative-error values at individual data points and query ranges; thus, for a given amount of synopsis space, they can offer tighter quality guarantees for reconstructed data values and approximate range aggregates than conventional deterministic wavelets.

For instance, consider our "bad" example array $A = [127, 71, 87, 31, 59, 3, 43, 99, 100, 42, 0, 58, 30, 88, 72, 130]$ used in the earlier sections of the article. Table VI depicts the original data values along with the values reconstructed from an 8-coefficient synopsis built using (a) deterministic, and (b) probabilistic thresholding using our MinRelVar and MinRelBias algorithms (with a sanity bound value of $\mathtt{S} = 5$).

The mean and maximum relative errors in data-value reconstruction (using the sanity bound of 5) for deterministic are 0.95 and 12.4, respectively; the corresponding numbers for MinRelVar and MinRelBias are only (0.31, 1.5) and (0.34, 1.5), respectively. Although no technique performs well on this pathological example, the numbers clearly show that our methods significantly improve both the mean and worst case relative error for individual data values.

As our results show, the most significant improvements occur with smaller synopses, larger skew, or noisy data. All experiments reported in this section were performed on a Sun Ultra-250 machine with 1 GB of main memory running Solaris 2.7.

4.1 Testbed and Methodology

4.1.1 *Techniques and Parameter Settings.* Our experimental study compares the conventional, deterministic thresholding scheme for Haar wavelet coefficients (i.e., maintaining the largest coefficients in absolute normalized value) with our three probabilistic thresholding schemes MinL2, MinRelVar, and MinRelBias (Section 3), designed to minimize the expected L^2 error, the maximum normalized standard error, and the maximum normalized bias, respectively.[8] For MinRelVar, we used the optimization described in Section 3.4, as it improved the accuracy of our answers.

For the normalizing methods (i.e., MinRelVar and MinRelBias), we determined a setting for the quantization parameter q used in our dynamic-programming solution by running our algorithm against a continuous mathematical-optimization solver for some small example data sets. Our results showed that our algorithm quickly converged to the optimal continuous solution even for relatively small values of the quantization parameter q. We decided to use a value of $\mathtt{q} = 10$ for our experimental runs, as we found that value to give good accuracy as well as reasonable running times for our dynamic-programming algorithm. For each input data set, we determined the value of the sanity bound S as the 10-percentile value in the data (i.e., 90% of the data points had values greater than S). Finally, we experimented with different values for the perturbation parameter Δ, and determined that $\Delta = \min\{0.01, \frac{\mathtt{S}}{100}\}$ was a good choice.

4.1.2 *Synthetic-Data and Range-Query Workload Generation.* We ran our techniques against several different one-dimensional synthetic data distributions, generated as follows. First, a Zipfian data generator was used to produce Zipfian frequencies for various levels of skew (controlled by the z parameter of the Zipfian), numbers of distinct values, and total frequency values (i.e., data-tuple counts). We varied the z parameter between 0.3 (low skew) to 2.0 (high skew), the distinct values between 128 and 512, and the tuple count between 100,000 and 500,000. Next, a permutation step was applied on the generated Zipfian frequencies to order them over the data domain; we experimented with four different permutation techniques: (1) *"NoPerm"* basically leaves the ordering as specified by the Zipfian data generator, that is, smaller values have higher frequencies; (2) *"Normal"* permutes the frequencies to resemble a bell-shaped normal distribution, with the higher (lower) frequencies at the center (respectively, ends) of the domain; (3) *"PipeOrgan"* permutes the frequencies in a "pipe-organ"-like arrangement, with higher (lower) frequencies at the two ends (respectively, center) of the data domain; and, (4) *"Random"* permutes the frequencies in a completely random manner over the data domain.

The range-aggregate queries in our test workloads were generated using different query range sizes varying between 10 and 100. For each given range size, we randomly generated 50 range-aggregate queries of that size by spreading the starting points of the query intervals in a uniform random fashion across the entire data domain. Our study included both SUM and AVERAGE aggregates computed over the query ranges in each workload.

4.1.3 *Approximation-Error Metrics.* We consider three metrics to gauge the accuracy of the different wavelet-synopsis techniques. Let d_i ($\hat{d}_i$) denote the ith accurate (respectively, reconstructed) value in the domain, and let S be the specified sanity bound for the approximation. The *maximum relative error*

[8]To the best of our knowledge, there are no known *deterministic* thresholding schemes that optimize for relative-error metrics. Furthermore, our dynamic-programming techniques do not directly extend to a deterministic setting; for example, our schemes rely on assigning fractional storage, $y_i \in (0, 1]$, to non-zero coefficients and then flipping coins to obtain $y_i \in \{0, 1\}$ (see Section 7).

in the data reconstruction is $\max_i\{\frac{|\hat{d}_i - d_i|}{\max\{d_i, S\}}\}$ The *mean relative error* is $\frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{d}_i - d_i|}{\max\{d_i, S\}}$. The *25-percentile relative error* is an upper bound on the relative error of 75% of the data values in the domain. As discussed in Section 3.7, the maximum relative error can be returned to the user as a *guaranteed-error bound* for the reconstruction of any individual data value, and our MinRelVar and MinRelBias techniques are designed to help minimize this error. However, it is based on only the largest error, and hence it provides a less informative comparison than the other two metrics. Thus, we will primarily use the mean relative error for the comparisons in this section (the 25-percentile relative error results are similar).

We also use the maximum, mean, and 25-percentile relative errors to compare the performance of the different wavelet-synopsis techniques over our range-aggregate query workloads. The definitions of these error metrics are essentially identical to those given above for data-value reconstruction, except that d_i ($\hat{d}_i$) now denote the accurate (respectively, approximate) answer to the ith range-aggregate query in our workload and N is the total number of such queries.

4.2 Experimental Results—Synthetic Data Sets

We present some of our experimental results with synthetic data sets for different frequency permutations and settings of Zipfian skew. The numbers shown in this section were obtained using a data domain of 256 distinct values and a tuple count of 200,000; we observed similar results for other parameter settings. We also focus on our empirical results for range-SUM aggregate query approximation as our results for range-AVERAGEs were qualitatively similar. After computing the y_i's for the MinL2, MinRelVar, and MinRelBias schemes, five trials of the coin flippings using different random seeds were performed, and the synopsis was selected that gave the least value for the observed mean relative error. Note that, due to the probabilistic nature of our thresholding schemes, there is always a (small) probability of a worst-case sequence of coin flips that could result in a poorly performing wavelet synopsis; thus, we recommend running our randomized procedures a small, constant number of times to ensure avoiding such worst-case scenarios.

4.2.1 *Data-Value Reconstruction Relative Errors: Mean, Maximum, and 25-Percentile.* Figure 6 depicts the data-reconstruction error numbers obtained by all four techniques for mean, maximum, and 25-percentile relative error on a "Normal" Zipfian data set with skew $z = 0.7$. Clearly, even for this moderate value of data skew, our MinRelVar and MinRelBias algorithms are able to guarantee better relative-error reconstruction for data values than deterministic, with the difference being especially evident for space-constrained synopses. More specifically, for 10 retained coefficients, both our methods give an over 200% improvement in mean relative error over deterministic, reducing it from over 0.5 to about 0.15. Remember that with 256 distinct values, 10 coefficients represent an approximately 4%-space synopsis of the full distribution. As the space for the synopsis is increased, the three methods converge to the

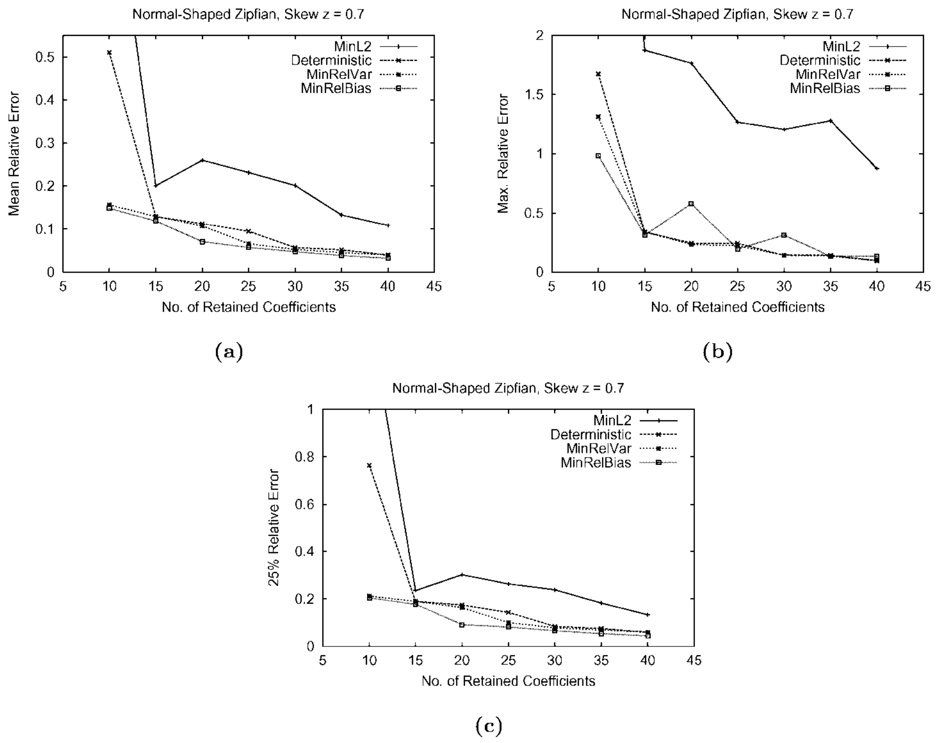
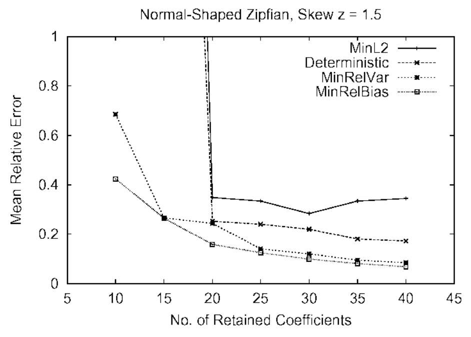


Fig. 6. Data-reconstruction approximation error for "Normal" Zipfian permutation with data skew $z = 0.7$: (a) Mean relative error. (b) Maximum relative error. (c) 25-percentile relative error.

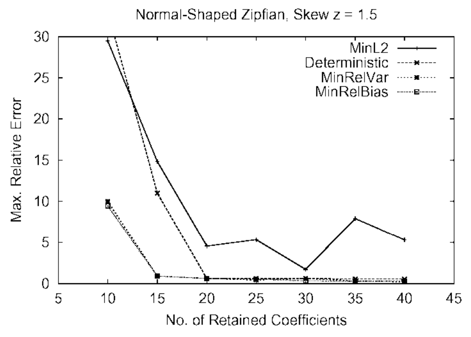
same relative error numbers. Similar trends can be seen for the maximum and 25-percentile relative errors. With respect to our MinL2 technique, our results show that it can result in large relative errors (even worse than those of deterministic); this is to be expected, because MinL2 does not explicitly optimize for the maximum normalized bias or variance in the reconstruction of data values.

Figure 7 shows the same set of plots obtained for a more skewed "Normal" Zipfian with $z = 1.5$. Again, we observe similar trends between the different strategies but, of course, the relative difference between deterministic and our MinRelBias and MinRelVar strategies is much more pronounced, with our methods offering improvements as high as 3,500% and 1,700% in mean relative error for 10 and 15 coefficients, respectively. The relative-error behavior of MinL2 is, once again, somewhat erratic; thus, we omit MinL2 from our discussion in the remainder of this section.

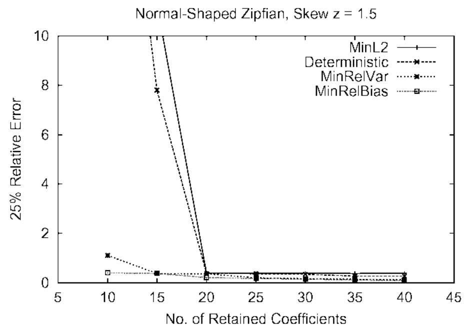
4.2.2 *Range-Aggregate Relative Errors: Mean, Maximum, and 25-Percentile.* Figure 8 depicts the range-SUM aggregate error numbers obtained by deterministic and our MinRelBias and MinRelVar techniques on a "Normal" Zipfian data set with skew $z = 1.0$ and a query range size of 30. Once again, it is evident that our probabilistic techniques are able to guarantee much lower relative-error numbers for approximate range aggregates, especially with space-constrained synopses. Our MinRelBias algorithm appears to be the consistent winner



(a)



(b)



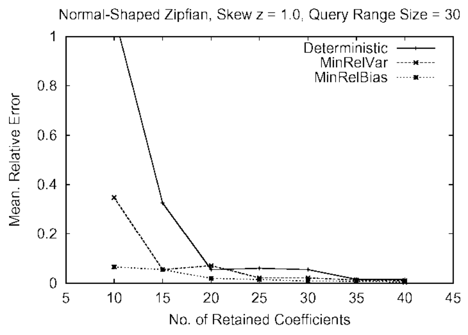
(c)

Fig. 7. Data-reconstruction approximation error for "Normal" Zipfian distribution with data skew $z = 1.5$: (a) Mean relative error. (b) Maximum relative error. (c) 25-percentile relative error.

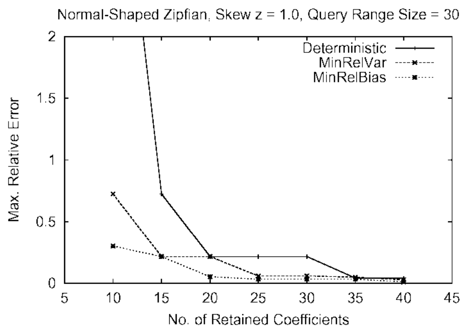
throughout the range of synopsis sizes, offering mean-relative error improvements in the range of 500–1,500% over deterministic thresholding for 10–15 coefficient synopses.

4.2.3 *Effect of Data Skew.* The plots in Figure 9 depict ratios between the mean relative error values obtained by deterministic and our MinRelBias and MinRelVar techniques for "Normal" Zipfian distributions with varying values of the skew parameter z. Obviously, the relative-error benefits obtained by our probabilistic techniques increase explosively as a function of the skew in the underlying data, with error ratios being way off the chart for $z = 1.5, 2.0$ and 10–15 coefficient synopses. For lower values of z, our probabilistic synopses offer more moderate benefits or match the relative-error performance of the deterministic scheme. It is also interesting to note that, even for higher synopsis sizes (e.g., 35–40 coefficients), our techniques can offer error improvements as high as 100% over deterministic thresholding.

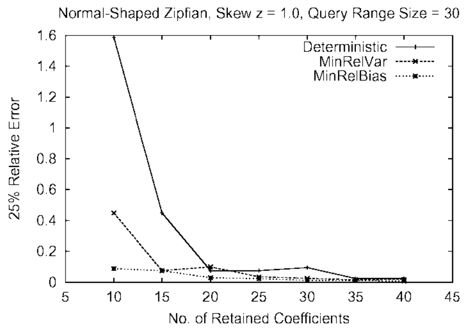
The plots in Figure 10 show the same mean relative error ratios for range-SUM aggregates with a range size of 20 over "Normal" Zipfian data for varying values of the data skew z. Once again, our MinRelBias and MinRelVar synopses consistently outperform or match (for lower values of z) conventional deterministic synopses in terms of relative error, with the difference being especially



(a)

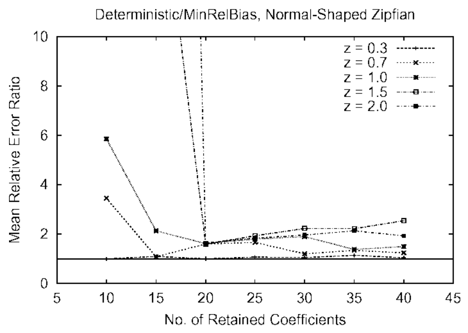


(b)

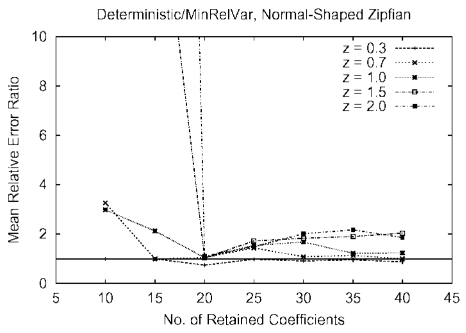


(c)

Fig. 8. Range-SUM aggregate approximation error for "Normal" Zipfian distribution with data skew $z = 1.0$ and query range size of 30: (a) Mean relative error. (b) Maximum relative Error. (c) 25-percentile relative error.

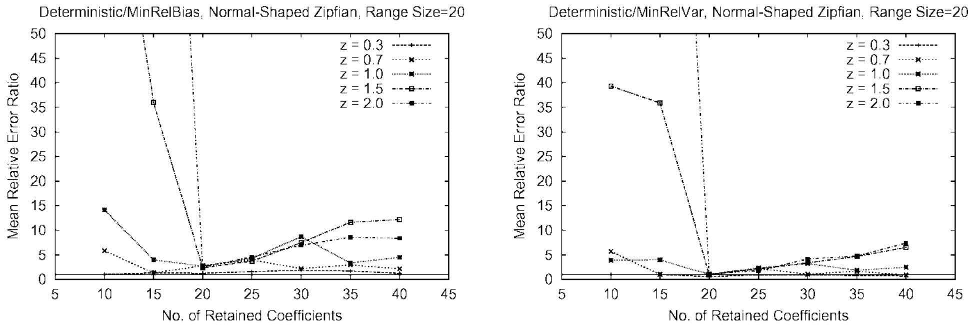


(a)



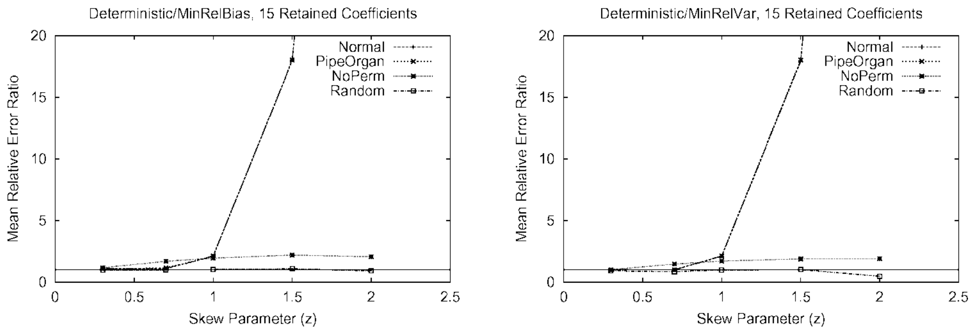
(b)

Fig. 9. Effect of Skew on Data Reconstruction: Ratio of mean relative reconstruction error between deterministic synopses and (a) MinRelBias synopses; (b) MinRelVar synopses. ("Normal" Zipfian distribution.)



(a) (b)

Fig. 10. Effect of Skew on Range-SUM Approximation: Ratio of mean relative aggregate error between deterministic synopses and (a) MinRelBias synopses; (b) MinRelVar synopses. ("Normal" Zipfian distribution, query range size = 20.)

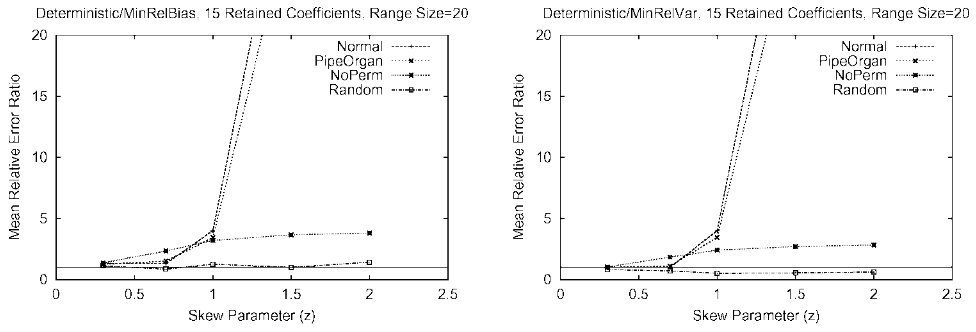


(a) (b)

Fig. 11. Effect of Permutation Strategy on Data Reconstruction: Ratio of mean relative reconstruction error between deterministic synopses and (a) MinRelBias synopses; (b) MinRelVar synopses. (15 retained coefficients.)

evident for smaller synopsis sizes and highly skewed data distributions. This, of course, is not surprising since, as the level of skew in the data increases, the number of large data frequencies (that the deterministic thresholding scheme tries to approximate well) becomes smaller and, thus, the number of range-SUM queries that they affect drops dramatically.

4.2.4 *Effect of Permutation Strategy.* Figure 11 shows the mean relative error ratios between deterministic and our MinRelBias and MinRelVar synopses as a function of the data skew parameter z for 15-coefficient synopses and for each of the four permutation strategies tested in our experiments. Clearly, the "Normal" and "PipeOrgan" frequency arrangements are the ones reaping the largest error benefits from our strategies, with the improvement increasing explosively for higher data skew; for example, for $z = 2.0$, both MinRelBias and MinRelVar reduce the mean relative error by over 8,000% with respect to deterministic. The error improvements for the non-permuted Zipfian ("NoPerm") are not as spectacular, but still are as high as 100–150% for more skewed



(a) (b)

Fig. 12. Effect of Permutation Strategy on Range-SUM Approximation: Ratio of mean relative aggregate error between deterministic synopses and (a) MinRelBias synopses; (b) MinRelVar synopses. (15 retained coefficients, query range size = 20.)

distributions. Finally, for the "Random" data set, our methods seem to closely match (in most cases) the performance of deterministic thresholding and, as a consequence, offer little (if any) benefit in terms of relative error. We should note, however, that by randomly permuting Zipfian frequencies, "Random" results in a highly irregular data set over which *any* small synopsis based on Haar wavelets is doomed to give poor approximations; thus, our results are not entirely surprising.

The plots in Figure 12 depict similar trends for the mean relative errors of our synopsis-construction techniques when approximating a workload of range-SUM aggregate queries with a range size of 20. As in the case of data reconstruction, our MinRelBias probabilistic-thresholding algorithm emerges as the consistent winner, offering significant relative-error improvements in the approximate range-SUM answers for three out of our four data-distribution shapes and (at least) matching the performance of deterministic thresholding on the highly irregular "Random" data set.

4.2.5 *Effect of Query Range Size.* Figure 13 depicts the mean relative error ratios between deterministic and our MinRelBias and MinRelVar techniques for 15-coefficient synopses and range-SUM aggregate query workloads over "Normal" Zipfian data as a function of the query range size for three different values of the data skew parameter z. Clearly, for skewed data, both our probabilistic techniques are able to guarantee significantly better mean relative errors for approximate range aggregates over the entire range of range sizes (10–100), with MinRelBias doing better than deterministic (by factors of up to 5) even for moderate data skew values ($z = 0.7$). (On the other hand, deterministic does as well as MinRelVar when $z = 0.7$.) Note that eventually, as the range size increases, the difference in relative error between our algorithms and deterministic thresholding starts decreasing as, with larger ranges, the large data frequencies (that deterministic tries to approximate accurately) start being included in a larger percentage of the queries in our workload. At the extreme case when the query range covers the entire data domain, all techniques are, of

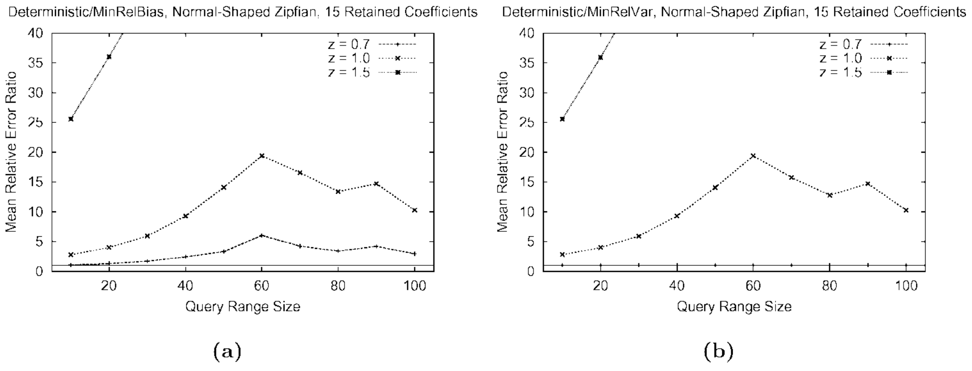


Fig. 13. Effect of Query Range Size on Range-SUM Approximation: Ratio of mean relative aggregate error between deterministic synopses and (a) MinRelBias synopses; (b) MinRelVar synopses. ("Normal" Zipfian distribution, 15 retained coefficients.)

course, bound to have zero error (assuming that they have all kept the overall average in the synopsis).

4.3 Experimental Results—Real-World Data Sets

To explore how our techniques performed on real-world data, we used the *Cover Type* data set from the National Forest Service, down-loaded from U.C. Irvine [Information and Computer Science, University of California at Irvine 2000]. There are 581,012 tuples in the data set; each tuple has 54 attributes (elevation, slope, distance to highway, forest cover type, etc). Most of these attributes have low cardinality, but there are 10 quantitative attributes, each with cardinality 256 or higher. We ran our techniques on a number of these attributes, and we report representative results on two of the attributes: "hillshade3pm" (CovType-HS3) and "aspect" (CovType-A). These two attributes test our algorithms under widely different distributions. CovType-HS3 measures a hillshade index (from 0 to 255) at 3pm on the summer solstice. Its histogram (the input to the synopsis techniques) is shown in Figure 14(a); as can be seen from the figure, the distribution is bell-shaped and relatively smooth. CovType-A measures the aspect in degrees azimuth, ranging from 0 to 359. Its histogram is shown in Figure 14(b); the distribution is more uniformly spread, with a pipe-organ-style fluctuation and considerable peaks of noise.

Figures 14(c,d) depict the ratio of mean relative errors between deterministic thresholding and our MinRelBias and MinRelVar schemes as the number of retained coefficients is varied. Clearly, our probabilistic wavelet synopses offer very substantial accuracy benefits over conventional deterministic synopses for both of the real-life data sets used in our tests. Further, our results show that for both CovType-HS3 and CovType-A the mean relative error numbers for the deterministic scheme improve very slowly as more space is given to the synopsis; thus, the relative performance of our schemes actually *improves* as the number of coefficients increases.

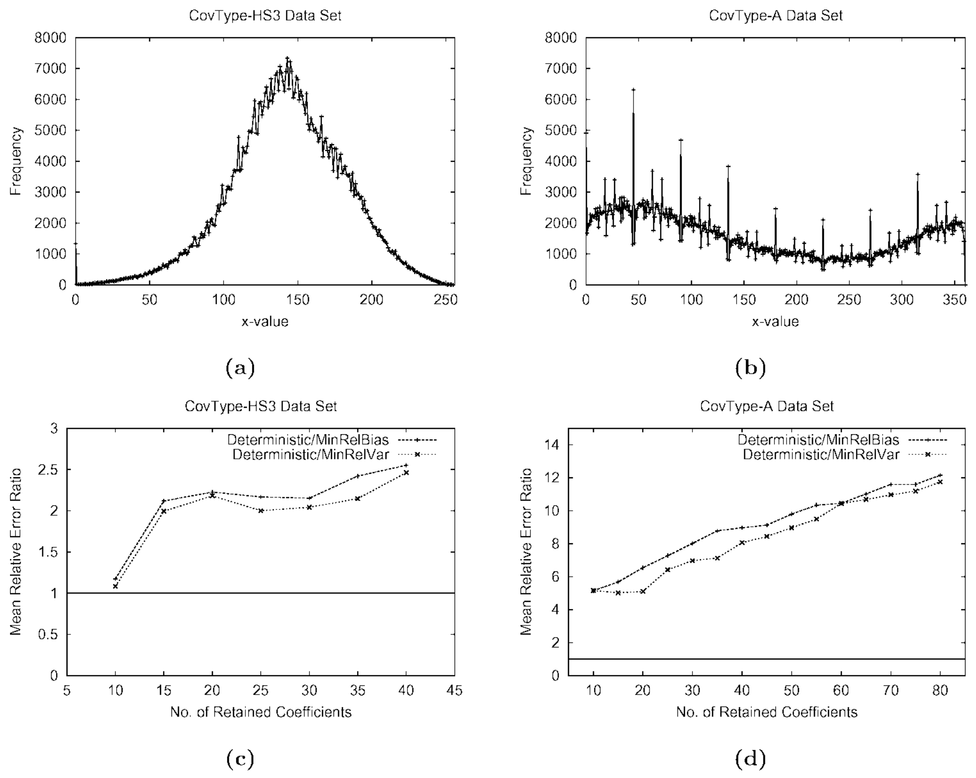


Fig. 14. (a,b) CovType-HS3 and CovType-A data sets. (c,d) Ratio of mean relative error between deterministic and probabilistic wavelet synopses for CovType-HS3 and CovType-A data.

5. EXTENSION TO MULTIDIMENSIONAL WAVELETS

In this section, we discuss the key ideas for extending our randomized rounding approach to multidimensional data. Dealing with multidimensional data sets raises two important issues that our techniques need to address. First, since recursive pairwise averaging and differencing steps are likely to significantly increase the density of the data in the wavelet-coefficient array, we *adaptively* threshold wavelet coefficients during the decomposition process; of course, the trick here is to ensure that this thresholding will not introduce any additional bias during data reconstruction. Second, we extend our basic dynamic-programming paradigm for randomized coefficient rounding so that it effectively works over the hierarchical error-tree structure for multidimensional data. This is the first wavelet-based compression technique for multidimensional data that provably enables unbiased reconstruction of data values and unbiased answers to range queries.

5.1 Adaptive Coefficient Thresholding during Decomposition

As observed in Chakrabarti et al. [2000], a nice feature of the nonstandard Haar decomposition is that the multidimensional wavelet-transform array W_A can be computed in one pass over the (suitably ordered) data array, with coefficients computed bottom up from the leaves of the error tree. Unfortunately, this pass may use far more than N_z space, where N_z is the number of nonzeros in the data array, because W_A can have far more than N_z nonzeros. Thus, as in Vitter and Wang [1999], we perform coefficient thresholding during the computation of W_A, to ensure that we retain at most N_z (nonzero) coefficients. We developed a new thresholding technique that performs this initial thresholding *without introducing any reconstruction bias* (previous schemes were deterministic and did not bound the errors they introduced during initial thresholding). Namely, in the one pass over the data array, we retain all coefficients computed until we reach our limit of N_z coefficients. At that point, we need to create room for more coefficients, so we select a threshold λ such that for say 20% of the coefficients c_i computed thus far, $|c_i| \le \lambda/2$. Then we flip coins, rounding each such c_i up to λ (or -λ, if $c_i < 0$) with probability $\frac{|c_i|}{\lambda}$ and down to zero with probability $1 - \frac{|c_i|}{\lambda}$. We expect to discard at least half of these coefficients, reducing the space by between 10% and 20%. Moreover, this does not introduce any reconstruction bias, because the expected value for each coefficient is still c_i. We then continue with the pass over the data array, repeating this process whenever we accumulate N_z coefficients. This results in a probabilistic wavelet synopsis $\mathtt{WS}_A$ with no more than N_z coefficients (and at least $\frac{4}{5} N_z$ coefficients).

The above captures the main idea, but the actual thresholding process we use is more complicated. First, we use $|c_i^*|$, the magnitude of the normalized coefficients, to select the pool of coefficients subject to coin flips. Second, each newly computed coefficient that would have been subjected to coin flips under the most recent thresholding, is subjected to this thresholding. Thus, at any point, all coefficients have been treated equally, independent of when they were computed. Third, each thresholding step with a new λ accounts for the previous thresholding steps in order to maintain unbiased answers and equal treatment. Fourth, since all coefficients are treated equally, we need not store the rounded value for retained coefficients, only the original value. (This property is used in the randomized-rounding phase described next.)

5.2 Rounding Multidimensional Coefficients using Dynamic Programming

As before, our goal is to have only $B \ll N_z$ coefficients. This we accomplish by extending our dynamic-programming schemes (e.g., MinRelVar) to select the appropriate rounding values y_i for multidimensional data distributions, and then flipping coins to reduce $\mathtt{WS}_A$ to B coefficients. We adapt the algorithm to account for the variance already incurred by rounded up values; this variance can be computed exactly because we have the original coefficient values c_i as well as the most recent value of λ used for adaptive thresholding. (We cannot account for the variance due to rounded down values, due to our space limit of N_z.)

Extending our dynamic-programming paradigm for wavelet-coefficient rounding to effectively handle multidimensional data is actually a nontrivial problem. This is because, even though the key “linearity” properties for reconstructing data values and range sums (Properties (P1) and (P2)) continue to hold for the multidimensional case, the underlying error-tree structure becomes

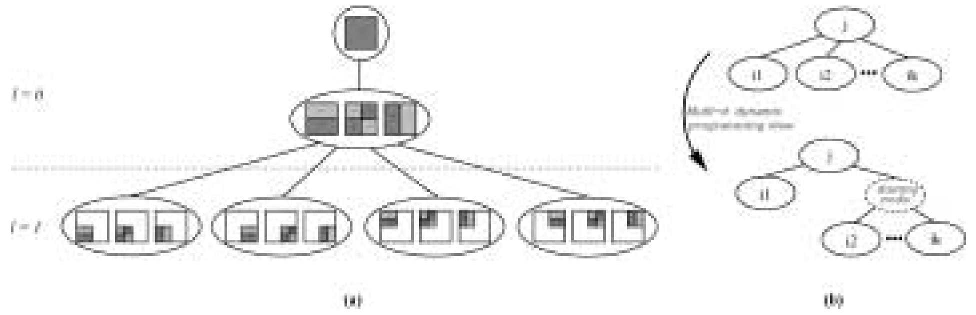


Fig. 15. (a) Error-tree structure for the sixteen nonstandard two-dimensional Haar coefficients for a 4×4 data array (actual data points are omitted for clarity). (b) Space allocation at node j: "ordered" view of our multidimensional dynamic-programming formulation.

significantly more complex. Remember that, in a d-dimensional error tree, each node (except for the root, i.e., the overall average) actually corresponds to a *set* of (at most) $2^d - 1$ wavelet coefficients that have exactly the same support region but different quadrant signs and magnitudes for their contribution. Furthermore, each (non-root) node t in a d-dimensional error tree can have up to 2^d children corresponding to the quadrants of the (common) support region of all coefficients in t.[9] As an example, Figure 15(a) depicts the error tree structure for the two-dimensional 4×4 Haar coefficient array in Figure 1(b). We now describe the details of our generalized, dynamic-programming scheme for randomized coefficient rounding that effectively deals with the above complications for multidimensional Haar wavelets. Our discussion focuses on the minimization of the maximum normalized standard error NSE (i.e., the multidimensional MinRelVar algorithm); the extensions to our other rounding algorithms following the same basic dynamic-programming paradigm are straightforward. We also center our description on formulating the key dynamic-programming recurrence using the hierarchical error-tree structure for Haar coefficients; our other technical ideas developed in Section 3.4 (e.g., the perturbation rule for zero coefficients and the solution-space quantization) are then applied as in the one-dimensional case.

Having (potentially) 2^d children at each internal node implies that a brute-force exploration of all the possible ways of distributing the space budget B among the children of the node (like the one used in our one-dimensional dynamic-programming recurrence (Eq. (12))) is no longer a feasible option. Given a quantization parameter q for the solution space, the time complexity of such a brute-force search at a single node with 2^d children would essentially be proportional to the number of ways of distributing qB balls among 2^d urns, which is $O((\mathrm{q}B)^{2^d-1})$. Instead, we propose a generalized dynamic-programming formulation for the multidimensional case that avoids this enormous search complexity at the cost of a slight increase in the amount of memory required to

[9]The number of children (coefficients) for an internal error-tree node can actually be less than 2^d (respectively, $2^d - 1$) when the sizes of the data dimensions are not all equal. In these situations, the exponent for 2 is determined by the number of dimensions that are *"active"* at the current level of the decomposition (i.e., those dimensions that are still being recursively split by averaging/differencing).

tabulate intermediate results. Specifically, let $M[\mathcal{J}, B]$, where $\mathcal{J} = \langle j_1, \ldots, j_n \rangle$ is a *list* of error-tree nodes, denote the optimal (i.e., minimum) value of the maximum $\textsc{nse}(\hat{d}_k)^2$ among all data values d_k belonging to any of the error subtrees in the list of subtrees $\langle T_{j_1}, \ldots, T_{j_n} \rangle$, assuming a total space budget of B. Thus, the cost of the optimal coefficient-rounding scheme with synopsis space of B is given by $M[\langle 0 \rangle, B]$. Also, let $\texttt{child}(j)$ denote the list of all child nodes of node j in some fixed, predetermined order and let $\texttt{coeff}(j)$ be the set of all wavelet coefficients corresponding to node j (we use $\texttt{coeff}(j) = \{0\}$ to denote that all these coefficients have zero values). Our multidimensional MinRelVar rounding scheme relies on the following dynamic-programming recurrence for $M[\mathcal{J}, B]$ (omitting the simpler case of the error-tree root, i.e., $\mathcal{J} = \langle 0 \rangle$):

$$M[\mathcal{J}, B] = \begin{cases} \displaystyle\min_{\substack{y_j \in (0, \min\{1, B\}];\\ b_L \in [0, B - y_j]}} \left\{ \max \left\{ \begin{array}{l} \frac{\mathrm{VAR}(j, y_j)}{\mathrm{NORM}(\langle i_1 \rangle)} + M[\langle i_1 \rangle, b_L], \\ \frac{\mathrm{VAR}(j, y_j)}{\mathrm{NORM}(\langle i_2, \ldots, i_k \rangle)} + M[\langle i_2, \ldots, i_k \rangle, B - y_j - b_L] \end{array} \right\} \right\}, \\ \qquad\qquad \text{if } \mathcal{J} = \langle j \rangle,\ \texttt{coeff}(j) \neq \{0\},\ \texttt{child}(j) = \langle i_1, \ldots, i_k \rangle, \text{ and } B > 0 \\ M[\langle i_1, \ldots, i_k \rangle, B] \quad \text{if } \mathcal{J} = \langle j \rangle,\ \texttt{coeff}(j) = \{0\}, \text{ and } \texttt{child}(j) = \langle i_1, \ldots, i_k \rangle \\ \min_{b_L \in [0, B]} \left\{ \max\{ M[\langle j_1 \rangle, b_L], M[\langle j_2, \ldots, j_n \rangle, B - b_L] \} \right\}, \\ \qquad\qquad \text{if } \quad \mathcal{J} = \langle j_1, \ldots, j_n \rangle, \text{ where } n > 1 \\ 0 \qquad\qquad \text{if } \quad \mathcal{J} = \phi \\ \infty \qquad\qquad \text{otherwise} \end{cases} \tag{20}$$

where $\textsc{norm}(\mathcal{J}) = \max\{\min_{j \in \mathcal{J}; k \in T_j} \{d_k^2\}, \mathtt{S}^2\}$ is the normalization term for the list of error subtrees $\langle T_j : j \in \mathcal{J} \rangle$. From the above dynamic-programming recurrence, it is easy to see that the values of the node-list argument $\mathcal{J}$ of M are not arbitrary lists of error-tree nodes: the only possible values for $\mathcal{J}$ are suffixes of the $\texttt{child}(j)$ lists for (nonleaf) nodes j in the error tree.

Intuitively, the recurrence in Eq. (20) generalizes that in Eq. (12) to nonbinary error-tree structures in which internal nodes can have any number of children. Furthermore, our generalized dynamic-programming formulation (based on *lists* of error-tree nodes) avoids the complexity of the brute-force search of possible space allocations to a node's children by *ordering* the search, based on some (fixed) ordering of the children nodes. Thus, for example, the first clause in Eq. (20) simply searches over all possible allotments of space y_j to a (nonzero) node j and, given a setting of y_j, over all possible allotments of the remaining $B - y_j$ budget amongst the *first* child subtree and the *list* of all remaining subtrees of node j. This ordering (also evident in the third clause of Eq. (20)) is depicted pictorially in Figure 15(b) and, as we demonstrate later in this section, results in an increase in time and space complexity of only $O(2^d)$ over the much simpler one-dimensional case. (Note that $O(2^d)$ is practically a constant factor for most reasonable values of the data dimensionality d: it is well known that data-reduction techniques based on space partitioning (like wavelets or histograms) become ineffective as d increases beyond 5–6 dimensions [Chakrabarti et al. 2000; Deshpande et al. 2001; Gunopulos et al. 2000].)

Our discussion thus far has not directly addressed the second complication introduced by multidimensional Haar wavelets, namely the fact that each node in a d-dimensional error tree actually represents a *set* of (up to) $2^d - 1$ distinct coefficients. This implies that, for a given allotment of space y_j to an error-tree node j, the contribution of that node to the overall path variance (i.e., the quantity $\text{VAR}(j, y_j)$ in Eq. (20)) must be computed based on the *optimal distribution* of y_j among all nonzero coefficients in $\texttt{coeff}(j)$. In other words, the node variance $\text{VAR}(j, y_j)$ is now the solution to the following minimization problem:

$$\text{Minimize} \sum_{c_i \in \texttt{coeff}(j) | c_i \neq 0} \text{Var}(C_i) = \sum_{c_i \in \texttt{coeff}(j) | c_i \neq 0} (\lambda_i - c_i) \cdot c_i$$

$$\text{subject to} \sum_{c_i \in \texttt{coeff}(j) | c_i \neq 0} \frac{c_i}{\lambda_i} \leq y_j \text{ and } \frac{c_i}{\lambda_i} \in (0, 1].$$

(Note that the space allotment y_j above can range anywhere between zero and the number of non-zero coefficients in $\texttt{coeff}(j)$.) Fortunately, this intranode optimization problem is very similar to our expected L^2 error minimization problem formulated in Section 3.3 and, as we have already shown, can be solved optimally in a very efficient manner by allocating each coefficient space $\frac{c_i}{\lambda_i}$ that is proportional to its magnitude $|c_i|$, while ensuring that all $\frac{c_i}{\lambda_i} \leq 1$. Based on the analysis of Section 3.3 and since each error-tree node can contain at most $2^d - 1$ coefficients, the time complexity of this extra intra-node optimization step is only $O(2^d \log 2^d) = O(d 2^d)$.

5.2.1 *Time and Space Complexity of Multidimensional Coefficient Rounding.* Assume a given quantization parameter q for the solution space of our multidimensional dynamic-programming recurrence. To estimate the time complexity of our d-dimensional coefficient rounding scheme, note that, assuming a fixed order on the child subtrees of each error-tree node, there are at most $O(N 2^d)$ possible choices for the node-list parameter $\mathcal{J}$ and a maximum of $\mathrm{q}B$ choices for the space allotment to each tree node. The time complexity of distributing a space budget of B among a list of (one or more) error subtrees is dominated by the first clause in Eq. (20), which runs in time $O(\mathrm{q} \log(\mathrm{q}B))$, for a given allotment B to the root of these subtrees (using a binary-search procedure similar to the one described in Section 3.4). Furthermore, solving the intranode space allocation problem at given node implies an additional cost of $O(d 2^d)$ bringing the overall complexity of processing an error-tree node for a given space allotment B to $O(\mathrm{q} \log(\mathrm{q}B) + d 2^d)$. (As always, computed optimal values are tabulated in the dynamic-programming array M for future reference.) Thus, the overall time complexity of d-dimensional coefficient rounding algorithm is $O(N 2^d \mathrm{q}B(\mathrm{q} \log(\mathrm{q}B) + d 2^d))$. The total space requirements of our d-dimensional algorithm are determined by the size of the dynamic-programming array M which is $O(N 2^d \mathrm{q}B)$; as in the one-dimensional case, however, we do not need to keep the full array M memory resident since the working-set size of our dynamic-programming scheme is only $O(2^d \mathrm{q} B \log N)$.

Even though the worst-case time and space complexity analysis outlined above uses N (i.e., the total number of cells in the data array), our multidimensional dynamic-programming algorithms can actually be implemented to run in overall time and space of $O(N_z 2^d \mathrm{q} B(\mathrm{q} \log(\mathrm{q}B) + d\,2^d))$ and $O(N_z 2^d \mathrm{q} B)$, respectively, where N_z is the number of nonzero coefficients retained. (Remember that, as discussed in Section 5.1, N_z is typically equal to the number of nonzero data cells, which can be much smaller than N for sparse multidimensional data.) The key observation here is that as our dynamic-programming schemes work in a bottom-up manner over the multidimensional error tree structure, the only situation in which we will have to incur computational cost at a tree node n with all zero coefficients (i.e., $\texttt{coeff}(n) = \{0\}$) is when n is the *least common ancestor* of at least two nonzero tree nodes beneath it in the tree. In this case, we have to apply our dynamic-programming recurrence at n to figure out how to best distribute the space allotment given to n across its nonzero descendants. (Note that if node n with $\texttt{coeff}(n) = \{0\}$ has just one nonzero descendant, then the M values computed at that descendant are just directly transferred to n.) Furthermore, once our algorithm has computed the M values for node n, its nonzero descendants can be discarded since they are no longer required for any computation that will take place further up in the error tree. Based on the above observations, it is easy to see that we can implement our algorithm so that the computation depicted in the dynamic-programming recurrence of Eq. (20) is performed for a total of at most $2N_z - 1$ error-tree nodes; that is, with $O(N_z)$ rather than $O(N)$ node computations. We should, of course, note that implementing our coefficient-rounding algorithm as described here may involve some additional computational overhead, for example, for sorting the N_z coefficients based on their inorder numbering in the error-tree and perhaps maintaining them in a heap to ensure that they are accessed in the appropriate “bottom-up” order. This overhead, however, is bounded by an additive factor of $O(N_z \log N_z)$ which would typically be dwarfed by the benefit of having $O(N_z)$ rather than $O(N)$ tree-node computations for sparse multidimensional data distributions.

6. RELATED WORK

Wavelets have a long history of successes in the signal and image processing arena [Jawerth and Sweldens 1994; Natsev et al. 1999; Stollnitz et al. 1996] and, recently, they have also found their way into data-management applications. Matias et al. [1998] first proposed the use of Haar-wavelet coefficients as synopses for accurately estimating the selectivities of range queries. Vitter and Wang [1999] describe I/O-efficient algorithms for building multidimensional 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. [2000] 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. Matias et al. [2000] consider the problem of on-line maintenance for coefficient synopses and propose a probabilistic-counting technique that approximately maintains the largest normalized-value coefficients in the presence of updates. Gilbert et al. [2001] propose algorithms for building approximate one-dimensional Haar-wavelet synopses over numeric data streams. All the above papers rely on conventional, deterministic thresholding schemes that typically decide the significance of a coefficient based on its absolute normalized value; hence, they suffer from the shortcomings described in this article.

There is a rich literature on m-term approximations using wavelets (m is the number of coefficients in the synopsis). Previous related work has studied dynamic-programming style approaches, deterministic thresholding to minimize a desired L^p metric, and bounds on worst-case error [DeVore 1998]. We are not aware of work addressing relative errors with sanity bounds, arguably the most important scenario for approximate query processing in databases. Anastassiou and Yu [1992a, 1992b] have written a series of papers on the topic of *probabilistic wavelet approximation*. However, these papers are unrelated to the approach we present, as they actually study the mathematical properties of certain wavelet operators for approximating continuous monotone functions and continuous probability distribution functions.

To the best of our knowledge, and after consulting several wavelet experts (e.g., D. Donoho [Personal communication 2001]), it seems that our approach of probabilistically rounding and selecting coefficients has not been previously studied.

7. DISCUSSION

In this section, we briefly discuss some issues related to the probabilistic wavelet synopses introduced in this article, and outline some challenging open problems and research directions.

The focus of our development thus far has been on the problem of building an error-optimal synopsis (e.g., minimizing the maximum relative reconstruction error) for a given amount of space. Our proposed techniques, however, are equally applicable to the *dual* version of this problem, namely minimizing the (expected) amount of space required by a probabilistic wavelet synopsis that achieves a given error guarantee. Determining such space-optimal, bounded-error synopses can be accomplished using our dynamic-programming techniques in conjunction with a *binary-search* procedure to find the optimal (smallest) required space budget B for the given error bound. Note that at most $O(\log N)$ search steps are necessary; furthermore, by the nature of our dynamic-programming solutions, the results obtained for smaller values of the space budget B can be re-used for larger space budgets $B' > B$, so that our time/space complexity bounds remain exactly the same with $B = B^*$, the optimal space budget.

A key property of our probabilistic wavelet synopses that enables them to outperform conventional wavelet synopses in our approximate query processing scenarios (see Section 4), is that they are specifically designed to optimize *relative-error metrics* (like normalized standard error or normalized bias) in the data reconstruction. This naturally raises the question of whether it would be possible to develop efficient, *deterministic* wavelet-thresholding schemes that are optimized for such relative-error metrics. As we already alluded to in Section 4, to the best of our knowledge, no such deterministic thresholding algorithms (for either relative or absolute value-reconstruction error) exist in the literature. Furthermore, our proposed techniques are intimately tied to the probabilistic problem formulations developed in this paper (in terms of both the resulting objective functions and the use of fractional storage for coefficients), and it is unclear whether they can be extended to a deterministic setting.

While it may be possible to design deterministic thresholding schemes for absolute or relative error metrics based on the resulting integer-programming formulations (e.g., using LP-relaxation and LP-rounding techniques [Vazirani 2001]), the resulting integer programs appear to be quite difficult to solve (or even approximate) in an efficient, scalable manner for large values of the domain size N. Developing effective (hopefully, combinatorial) deterministic wavelet-thresholding algorithms for relative-error metrics and comparing them against the probabilistic solutions proposed in this article is certainly a challenging direction for future research in this area. Another important question in this realm concerns the suitability of the Haar wavelet transform as a data-summarization and approximate query processing tool when it comes to error metrics other than L^2. Could there be other (existing or new) wavelet-basis functions that are better suited for optimizing relative error metrics in the data approximation?

Finally, an important practical issue not addressed in this article is that of *incremental maintenance* of our probabilistic wavelet synopses in a dynamic (e.g., data-warehousing) environment. A key assumption underlying all our proposed techniques is that the underlying data set is given and static. Incrementally maintaining a probabilistic synopsis of B wavelet coefficients (or, their rounded values) so that it retains its optimality (e.g., with respect to normalized standard error) over a dynamically-updated data set is a difficult open problem. Note that, unlike conventional coefficient thresholding (where the goal is to simply keep track of the top-B (normalized) coefficient values), an incremental-maintenance technique for our synopses would need to continuously track the solution of a rather complex optimization problem over the changing data distribution. It is unclear whether the probabilistic-counting ideas of Matias et al. [2000] would be applicable or even useful in this setting.

8. CONCLUSIONS

This article has introduced *probabilistic wavelet synopses*, the first wavelet-based data reduction technique optimized for guaranteed accuracy of individual approximate answers. Our technique enables unbiased or low-bias data reconstruction. We have described a number of novel techniques for tuning our scheme to minimize desired error metrics, as well as extensions to multidimensional data. Experimental results on real-world and synthetic data sets demonstrate that probabilistic wavelet synopses significantly reduce approximation relative error compared with the previous deterministic approach, in most cases by over a factor of 2, and up to a factor of 80 for highly skewed data. Therefore, we recommend their use in approximate query processing systems.

ACKNOWLEDGMENTS

The authors thank the anonymous referees for insightful comments on the article.

REFERENCES

ACHARYA, S., GIBBONS, P. B., POOSALA, V., AND RAMASWAMY, S. 1999. Join synopses for approximate query answering. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data* (Philadelphia, Pa.). ACM, New York, 275–286.

AMSALEG, L., BONNET, P., FRANKLIN, M. J., TOMASIC, A., AND URHAN, T. 1997. Improving responsiveness for wide-area data access. *IEEE Data Eng. Bull. 20*, 3 (Sept.), 3–11. (Special Issue on Improving Query Responsiveness).

ANASTASSIOU, G. A. AND YU, X. M. 1992a. Monotone and probabilistic wavelet approximation. *Stoch. Anal. Appl. 10*, 3, 251–264.

ANASTASSIOU, G. A. AND YU, X. M. 1992b. Probabilistic discrete wavelet approximation. *Num. Func. Anal. Opt. 13*, 117–121.

CHAKRABARTI, K., GAROFALAKIS, M., RASTOGI, R., AND SHIM, K. 2000. Approximate query processing using wavelets. In *Proceedings of the 26th International Conference on Very Large Data Bases* (Cairo, Egypt). 111–122.

DESHPANDE, A., GAROFALAKIS, M., AND RASTOGI, R. 2001. Independence is good: Dependency-based histogram synopses for high-dimensional data. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data* (Santa Barbara, Calif.). ACM, New York, 199–210.

DEVORE, R. A. 1998. Nonlinear approximation. *Acta Numer. 7*, 51–150.

GILBERT, A. C., KOTIDIS, Y., MUTHUKRISHNAN, S., AND STRAUSS, M. J. 2001. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proceedings of the 27th International Conference on Very Large Data Bases* (Rome, Italy). 79–88.

GUNOPULOS, D., KOLLIOS, G., TSOTRAS, V. J., AND DOMENICONI, C. 2000. Approximating multidimensional aggregate range queries over real attributes. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (Dallas, Tex.). ACM, New York, 463–474.

HAAS, P. J. AND SWAMI, A. N. 1992. Sequential sampling procedures for query size estimation. In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data* (San Diego, Calif.). ACM, New York, 341–350.

HELLERSTEIN, J. M., HAAS, P. J., AND WANG, H. J. 1997. Online aggregation. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data* (Tucson, Az.). ACM, New York, 171–182.

INFORMATION AND COMPUTER SCIENCE, UNIVERSITY OF CALIFORNIA AT IRVINE. 2000. ftp://ftp.ics.uci.edu/pub/machine-learning-databases.

JAWERTH, B. AND SWELDENS, W. 1994. An overview of wavelet based multiresolution analyses. *SIAM Rev. 36*, 3, 377–412.

MATIAS, Y. 1992. Highly parallel randomized algorithmics. Ph.D. dissertation, Tel-Aviv Univ. Tel-Aviv, Israel.

MATIAS, Y., VITTER, J. S., AND WANG, M. 1998. Wavelet-based histograms for selectivity estimation. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data* (Seattle, Wash.). ACM New York, 448–459.

MATIAS, Y., VITTER, J. S., AND WANG, M. 2000. Dynamic maintenance of wavelet-based histograms. In *Proceedings of the 26th International Conference on Very Large Data Bases* (Cairo, Egypt). 101–110.

MOTWANI, R. AND RAGHAVAN, P. 1995. *Randomized Algorithms*. Cambridge University Press, Cambridge, Mass.

NATSEV, A., RASTOGI, R., AND SHIM, K. 1999. WALRUS: A similarity retrieval algorithm for image databases. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data* (Philadelphia, Pa.). ACM, New York, 395–406.

RAGHAVAN, P. AND THOMPSON, C. 1987. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica 7*, 4, 365–374.

STOLLNITZ, E. J., DEROSE, T. D., AND SALESIN, D. H. 1996. *Wavelets for Computer Graphics—Theory and Applications*. Morgan-Kaufmann, San Francisco, Calif.

VAZIRANI, V. V. 2001. *Approximation Algorithms*. Springer-Verlag, New York.

VITTER, J. S. AND WANG, M. 1999. Approximate computation of multidimensional aggregates of sparse data using wavelets. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data* (Philadelphia, Pa.). ACM, New York, 193–204.

Received September 2002; revised July 2003; accepted August 2003