Deterministic Wavelet Thresholding for Maximum-Error Metrics

Minos Garofalakis
Bell Laboratories
Lucent Technologies
600 Mountain Avenue
Murray Hill, NJ 07974
minos@research.bell-labs.com

Amit Kumar
Dept. of Computer Science and Engineering
Indian Institute of Technology, Delhi
Hauz Khas, New Delhi 110 016, India
amitk@cse.iitd.ernet.in

ABSTRACT

Several studies have demonstrated the effectiveness of the wavelet decomposition as a tool for reducing large amounts of data down to compact *wavelet synopses* that can be used to obtain fast, accurate approximate answers to user queries. While conventional wavelet synopses are based on greedily minimizing the overall root-mean-squared (i.e., L_2-norm) error in the data approximation, recent work has demonstrated that such synopses can suffer from important problems, including severe bias and wide variance in the quality of the data reconstruction, and lack of non-trivial guarantees for individual approximate answers. As a result, *probabilistic thresholding schemes* have been recently proposed as a means of building wavelet synopses that try to *probabilistically* control other approximation-error metrics, such as the maximum relative error in data-value reconstruction, which is arguably the most important for approximate query answers and meaningful error guarantees.

One of the main open problems posed by this earlier work is whether it is possible to design efficient *deterministic* wavelet-thresholding algorithms for minimizing non-L_2 error metrics that are relevant to approximate query processing systems, such as maximum relative or maximum absolute error. Obviously, such algorithms can guarantee better wavelet synopses and avoid the pitfalls of probabilistic techniques (e.g., "bad" coin-flip sequences) leading to poor solutions. In this paper, we address this problem and propose novel, computationally efficient schemes for deterministic wavelet thresholding with the objective of optimizing maximum-error metrics. We introduce an *optimal* low polynomial-time algorithm for *one-dimensional* wavelet thresholding – our algorithm is based on a new Dynamic-Programming (DP) formulation, and can be employed to minimize the maximum relative or absolute error in the data reconstruction. Unfortunately, directly extending our one-dimensional DP algorithm to *multi-dimensional* wavelets results in a super-exponential increase in time complexity with the data dimensionality. Thus, we also introduce novel, polynomial-time *approximation schemes* (with tunable approximation guarantees for the target maximum-error metric) for deterministic wavelet thresholding in multiple dimensions.

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 [13]. 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 [2]. 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) [1].

Wavelets provide a mathematical tool for the hierarchical decomposition of functions, with a long history of successful applications in signal and image processing [14, 19, 20]. Recent studies have also demonstrated the applicability of wavelets to selectivity estimation [15] and to approximate query processing over massive relational tables [3, 21] and data streams [16, 10]. 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 Matias, Vitter, and Wang [15, 21] and Chakrabarti et al. [3] have demonstrated that fast and accurate approximate query processing engines can be designed to operate solely over such compact *wavelet synopses*.

A major shortcoming of conventional wavelet-based techniques for approximate query processing (including all the above-cited studies) is the fact that the quality of approximate answers can vary widely and no meaningful error guarantees can be provided to the users of the approximate query engine. Coefficients in conventional wavelet synopses are typically chosen to optimize the overall root-mean-squared (i.e., L_2-norm average) error in the data approximation which, as demonstrated in a recent study by Garofalakis and Gibbons [7, 8], can result in wide variance as well as severe bias in the quality of the approximation over the underlying domain of data values. Their proposed solution, termed *probabilistic wavelet synopses*, relies on probabilistic-thresholding schemes (based on randomized rounding [17]) for synopsis construction that try to *probabilistically* control other approximation-error metrics, such as the *maximum relative error* in the data reconstruction [7, 8]. Such maximum relative-error metrics are arguably the most important for effective approximate query processing and can provide meaningful error guarantees for individual approximate answers. In order to probabilistically control maximum relative error, the algorithms proposed in [7, 8] explicitly try to minimize appropriate probabilistic metrics (such as normalized standard error or normalized bias) for the randomized synopsis construction process [7, 8]. (Similar schemes are also given for controlling maximum *absolute* error.)

Our Contributions. A potential problem with the probabilistic-thresholding techniques of [7, 8] is that, exactly due to their probabilistic nature, there is always a possibility of a "bad" sequence of coin flips resulting in a poor synopsis; furthermore, they are based on a *quantization* of the possible synopsis-space allotments, whose impact on the quality of the final synopsis is not entirely clear. Clearly, a *deterministic* thresholding algorithm that explicitly minimizes the relevant maximum-error metric (e.g., maximum relative error) in the synopsis, is always guaranteed to give better results. Unfortunately, as already pointed out by by Garofalakis and Gibbons [7, 8], their thresholding algorithms depend critically on the probabilistic nature of their solution, and are inapplicable in a deterministic setting. In fact, one of the main open problems cited in [7, 8] is whether it is possible to design efficient deterministic thresholding for minimizing non-L_2 error metrics that are relevant for approximate query answers, such as the maximum relative or absolute error in the data approximation.

In this paper, we propose novel, computationally efficient schemes for deterministic wavelet thresholding with the objective of optimizing maximum-error metrics (e.g., maximum relative error). We start by presenting an *optimal* low polynomial-time algorithm for *one-dimensional* wavelet thresholding. Our algorithm is based on a novel Dynamic-Programming (DP) formulation and can be employed to minimize either maximum relative error or maximum absolute error in the data reconstruction – its running time and working-space requirements are only $O(N^2\ B \log B)$ and $O(NB)$, respectively, where N denotes the size of the data domain and B is the desired size of the synopsis (i.e., number of retained coefficients). Unfortunately, directly extending our optimal DP algorithm to *multi-dimensional* wavelets results in a super-exponential increase in time complexity with the data dimensionality, rendering such a solution unusable even for the relatively small dimensionalities where wavelets are typically used (e.g., 2–5 dimensions). Thus, we also introduce two efficient, polynomial-time *approximation schemes* (with tunable ϵ-approximation guarantees for the target maximum-error metric) for deterministic wavelet thresholding in multiple dimensions. Both our approximation schemes are based on approximate dynamic programs that tabulate a much smaller number of sub-problems than the optimal DP solution, while guaranteeing a small deviation from the optimal objective value. More specifically, our first approximation algorithm can give ϵ-additive-error guarantees for maximum relative or absolute error, whereas our second algorithm is a $(1+\epsilon)$-approximation scheme for maximum absolute error – the running time for both our approximation schemes is roughly proportional to $O(\frac{1}{\epsilon}\ N \log^2 N\ B \log B)$. To the best of our knowledge, our work is the *first* to propose efficient optimal and near-optimal algorithms for building wavelet synopses optimized for maximum-error metrics in one or multiple dimensions.

Organization. The remainder of this paper is organized as follows. Section 2 discusses background material on the wavelet decomposition and wavelet data synopses. Then, in Section 3, we develop our deterministic maximum-error thresholding algorithms for both one- and multi-dimensional wavelets. Section 4 gives an overview of related work and, finally, Section 5 outlines our conclusions along with some interesting directions for future research in this area.

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 [20]. 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 $N = 8$ data values $A = [2, 2, 0, 2, 3, 5, 4, 4]$. 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 $[2, 1, 4, 4]$. In other words, the average of the first two values (that is, 2 and 2) is 2, that of the next two values (that is, 0 and 2) is 1, 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 0 since $2 - 2 = 0$, for the second we again need to store -1 since $1 - 2 = -1$. Note that no information has been lost in this process – it is fairly simple to reconstruct the eight values of the original data array from the lower-resolution array containing the four averages and the four detail coefficients. Recursively applying the above pairwise averaging and differencing process on the lower-resolution array containing the averages, we get the following full decomposition:

Resolution	Averages	Detail Coefficients
3	[2, 2, 0, 2, 3, 5, 4, 4]	—
2	[2, 1, 4, 4]	[0, -1, -1, 0]
1	[3/2, 4]	[1/2, 0]
0	[11/4]	[-5/4]

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 $W_A = [11/4,\ -5/4,\ 1/2,\ 0,\ 0,\ -1,\ -1,\ 0]$. Each entry in W_A is called a *wavelet coefficient*. The main advantage of using W_A instead of the original data vector A is that for vectors containing similar values most of the detail coefficients tend to have very small values. Thus, eliminating such small coefficients from the wavelet transform (i.e., treating them as zeros) introduces only small errors when reconstructing the original data, resulting in a very effective form of lossy data compression [20].

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 [20] 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). Thus, the normalized coefficient, c_i^*, is $c_i/\sqrt{2^{\texttt{level}(c_i)}}$.

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 [15]. 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(a) depicts the error tree for our simple example data vector A. Each internal node c_i ($i = 0, \ldots, 7$) is associated with a wavelet coefficient value, and each leaf d_i ($i = 0, \ldots, 7$) 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", "coefficient", and "node/coefficient value" interchangeably in what follows. For ease of reference, Table 1 summarizes some of the key notation used in this paper 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 – extensions to multi-dimensional wavelets are straightforward. Additional notation will be introduced when necessary.

Symbol $i \in \{0..N-1\}$	Description
N	Number of data-array cells
D	Data-array dimensionality
B	Space budget for synopsis
A, W_A	Input data and wavelet transform arrays
d_i	Data value for i^{th} data-array cell
$\hat{d}_i$	Reconstructed data value for i^{th} array cell
c_i	Haar coefficient at index/coordinate i
T_i	Error subtree rooted at node c_i
$\texttt{coeff}(T_i), \texttt{data}(T_i)$	Coefficient/data values in T_i subtree
$\texttt{path}(u)$	All non-zero proper ancestors of node u in the error tree
s	Sanity bound for relative-error metric
$\mathsf{relErr}_i, \mathsf{absErr}_i$	Relative/absolute error for data value d_i

Table 1: Notation.

Given a node u in an error tree T, let $\texttt{path}(u)$ denote the set of all proper ancestors of u in T (i.e., the nodes on the path from u to the root of T, including the root but not u) with non-zero coefficients. A key property of the Haar wavelet decomposition is that the reconstruction of any data value d_i depends only on the values of coefficients on $\texttt{path}(d_i)$; more specifically, we have

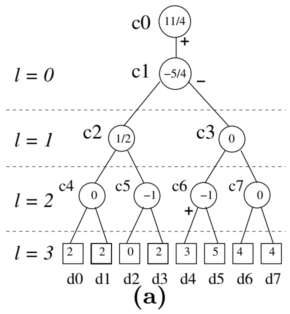
$$d_i = \sum_{c_j \in \texttt{path}(d_i)} \mathsf{sign}_{ij} \cdot c_j, \qquad (1)$$

where $\mathsf{sign}_{ij} = +1$ if d_i is in the left child subtree of c_j or $j = 0$, and $\mathsf{sign}_{ij} = -1$ otherwise. Thus, reconstructing any data value involves summing at most $\log N + 1$ coefficients. For example, in Figure 1(a), $d_4 = c_0 - c_1 + c_6 = \frac{11}{4} - (-\frac{5}{4}) + (-1) = 3$. The *support region* for a coefficient c_i is defined as the set of (contiguous) data values that c_i is used to reconstruct; the support region for a coefficient c_i is uniquely identified by its coordinate i.

2.2 Multi-Dimensional Haar Wavelets

The Haar wavelet decomposition can be extended to *multi-dimensional* data arrays using two distinct methods, namely the *standard* and *nonstandard* Haar decomposition [20]. 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 [3, 21].

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 [3, 21].) 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.



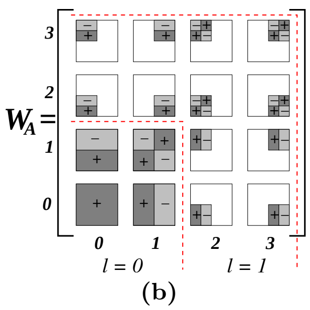


Figure 1: (a) Error-tree structure for our example data array A ($N = 8$). (b) Support regions and signs for the sixteen nonstandard two-dimensional Haar basis functions. The coefficient magnitudes are multiplied by $+1$ (-1) where a sign of $+$ (resp., $-$) appears, and 0 in blank areas.

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, i.e., 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 [3, 20].

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, but their semantics and structure are somewhat more complex. 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 and magnitudes for their contribution. Furthermore, each (non-root) 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.[1] (Note that the sign of each coefficient's contribution to the leaf (data) values residing at each of its children in the tree is determined by the coefficient's quadrant sign information.) As an example, Figure 2 depicts the error-tree structure for the two-dimensional 4×4 Haar coefficient array in Figure 1(b).

[1]The number of children (coefficients) for an internal error-tree node can actually be less than 2^D (resp., $2^D - 1$) 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).

Thus, the (single) child t of the root node contains the coefficients $W_A[0, 1]$, $W_A[1, 0]$, and $W_A[1, 1]$, and 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.

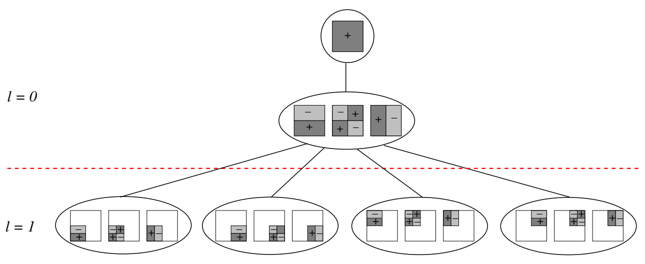


Figure 2: Error-tree structure for the sixteen nonstandard two-dimensional Haar coefficients for a 4×4 data array (data values omitted for clarity).

Based on the above generalization of the error-tree structure to multiple dimensions, we can naturally extend the formula for data-value reconstruction (Equation (1)) to multi-dimensional Haar wavelets. Once again, the reconstruction of d_i depends only on the *coefficient sets* for all error-tree nodes in $\texttt{path}(d_i)$, where the sign of the contribution for each coefficient W in node $t \in \texttt{path}(d_i)$ is determined by the quadrant sign information for W.

2.3 Wavelet-Based Data Reduction: Coefficient Thresholding

Given a limited amount of storage for building a *wavelet synopsis* of the input data array A, a thresholding procedure retains a certain number $B << N$ of the coefficients in W_A as a highly-compressed approximate representation of the original data (the remaining coefficients are implicitly set to 0). 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. The method of choice for the vast majority of earlier studies on wavelet-based data reduction and approximation [3, 15, 16, 21] is *conventional coefficient thresholding* that greedily retains retains the B largest Haar-wavelet coefficients in *absolute normalized value*. It is a well-known fact that 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 [20]. 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).

Unfortunately, wavelet synopses optimized for overall L_2 error may not always be the best choice for approximate query processing systems. As observed in the recent study of Garofalakis and Gibbons [7, 8], such conventional wavelet synopses suffer from several important problems, including the introduction of severe bias in the data reconstruction and wide variance in the quality of the data approximation, as well as the lack of non-trivial guarantees for individual approximate answers. To address these shortcomings, their work introduces novel, *probabilistic* thresholding schemes based on randomized rounding [17], that probabilistically round coefficients either up to a larger rounding value (to be retained in the synopsis) or down to zero. Intuitively, their probabilistic schemes assign each non-zero coefficient *fractional storage* $y \in (0, 1]$ equal to its retention probability, and then flip independent, appropriately-biased coins to construct the synopsis. Their winning strategies (termed MinRelVar and MinRelBias in [7, 8]) are based on trying to *probabilistically* control the the *maximum relative error* (with an appropriate *sanity bound* s)[2] in the approximation of individual data values based on the synopsis; that is, they attempt to minimize the quantity

$$\max_i \left\{ \frac{|\hat{d}_i - d_i|}{\max\{|d_i|, \mathsf{s}\}} \right\},$$

(where $\hat{d}_i$ denotes the data value reconstructed based on the synopsis) with sufficiently high probability. More specifically, their MinRelVar and MinRelBias algorithms are based on *Dynamic-Programming (DP)* formulations that explicitly minimize appropriate probabilistic metrics, such as the maximum normalized standard error (MinRelVar) or the maximum normalized bias (MinRelBias), in the randomized synopsis construction; these formulations are then combined with a *quantization* of the potential fractional-storage allotments to give combinatorial thresholding algorithms [7, 8]. (Similar probabilistic schemes are also given for probabilistically controlling maximum *absolute* error.)

3. OUR SOLUTION: DETERMINISTIC WAVELET THRESHOLDING

Rather than trying to probabilistically control maximum relative error through the optimization of probabilistic measures (like normalized standard error or bias [7, 8]), a more direct solution would be to design *deterministic* thresholding schemes that explicitly minimize maximum-error metrics. Obviously, such schemes can not only guarantee better synopses, but they can also avoid the potential pitfalls of randomized techniques, as well as the space-quantization requirement of [7, 8] whose impact on the quality of the final solution is not entirely clear. Unfortunately, as already pointed out by Garofalakis and Gibbons [8], their DP formulations and algorithms depend crucially on the ability to assign *fractional storage* (i.e., retention probability) values in $(0, 1]$ to individual coefficients, which renders their schemes inapplicable for *deterministic* wavelet thresholding (where the storage assigned to each coefficient is either 0 or 1). In fact, one of the main open problems in [7, 8] is whether it is possible to design efficient algorithms for *deterministic* Haar-coefficient thresholding for minimizing non-L_2 error metrics that are relevant for approximate query answers, such as maximum relative or absolute error in the data approximation. We now attack this problem for both one- and multi-dimensional Haar wavelets.

3.1 One-Dimensional Wavelet Thresholding

In this section, we propose a novel, simple, low polynomial-time scheme based on Dynamic-Programming (DP) for building *deterministic* Haar-wavelet synopses that minimize the *maximum relative or absolute error* in the data-value approximation. More formally, let $\mathsf{relErr}_i = \frac{|\hat{d}_i - d_i|}{\max\{|d_i|, \mathsf{s}\}}$ be an error metric that combines relative error with an appropriate sanity bound s (as in [7, 8]) and, similarly, let $\mathsf{absErr}_i = |\hat{d}_i - d_i|$ denote the absolute approximation error for the i^{th} data value. Our deterministic wavelet thresholding problem can be formally defined as follows.

[Deterministic Maximum Relative/Absolute Error Minimization] Given a synopsis space budget B, determine a subset of at most B Haar-wavelet coefficients that minimize the maximum relative (or, absolute) error in the data-value approximation; that is, for $\mathsf{err} \in \{\mathsf{relErr}, \mathsf{absErr}\}$, *minimize* $\max_{i \in \{0,\ldots,N-1\}}\{\mathsf{err}_i\}$ ∎

The important thing to note here is that, unlike the probabilistic normalized standard error or normalized bias metrics employed in [7, 8], our relErr and absErr metrics do not have a simple monotonic/additive structure over the Haar-coefficient error tree. The key problem, of course, is that, even though data values are simple linear combinations of coefficients, each coefficient contributes with a *different sign* on different parts of the underlying data domain. Unfortunately, the above facts also imply that the DP formulations in [7, 8] are no longer applicable, since the assumed "principle of optimality" for error subtrees is no longer valid; in other words, due to the absence of additive/monotonic structure for the objective, the optimal solution for the subtree rooted at node c_j is not necessarily a combination of the optimal (partial) solutions for c_j's child subtrees.

We now formulate a novel DP recurrence and algorithm for our deterministic maximum-error optimization problem. In a nutshell, the basic idea in our new DP formulation is to condition the optimal error value for an error subtree not only on the root node c_j of the subtree and the amount B of synopsis storage allotted, but also on *the error that "enters" that subtree* through the coefficient selections made on the path from the root to node c_j (excluding c_j itself), i.e., coefficient selections on $\mathtt{path}(c_j)$. The key observation here is that, since the depth of the error tree is $O(\log N)$, we can afford to tabulate all such possible selections while keeping the running-time of our algorithm in the low-polynomial range.

More formally, let B denote the total space budget for the synopsis, and let T_j be the subtree of the error-tree rooted at node c_j, with $\mathtt{coeff}(T_j)$ ($\mathtt{data}(T_j)$) denoting the set of coefficient (resp., data) values in T_j. Finally, let $M[j, b, S]$ denote the optimal (i.e., minimum) value of the maximum error (relative or absolute) among all data values in T_j assuming a synopsis space budget of b coefficients for the T_j subtree, *and* that a subset $S \subseteq \mathtt{path}(c_j)$ (of size at most $\min\{B - b, \log N + 1\}$) of proper ancestors of c_j have been selected for the synopsis; that is, assuming a relative-error metric (i.e., $\mathsf{err} = \mathsf{relErr}$),

$$M[j, b, S] = \min_{S_j \subseteq \mathtt{coeff}(T_j), |S_j| \le b} \left\{ \max_{d_i \in \mathtt{data}(T_j)} \mathsf{relErr}_i \right\},$$

where

$$\mathsf{relErr}_i = \frac{|d_i - \sum_{c_k \in \mathtt{path}(d_i) \cap (S_j \cup S)} \mathsf{sign}_{ik} \cdot c_k|}{\max\{|d_i|, \mathsf{s}\}}.$$

[2]The role of the sanity bound is to ensure that relative-error numbers are not unduly dominated by small data values [12, 21].

(The case for absolute error (i.e., $\text{err} = \text{absErr}$) is defined similarly.) We now formulate a DP recurrence for computing the $M[j, b, S]$ entries; clearly, $M[0, B, \phi]$ gives us the desired optimal error value at the root node of the error tree (the corresponding error-optimal wavelet synopsis can then be built by simply re-tracing the choices of our DP computation using standard techniques).

The *base case* for our recurrence occurs for data (i.e., leaf) nodes in the Haar error tree; that is, for $c_j = d_{j-N}$ with $j \geq N$ (see Figure 1(a)). In this case, $M[j, b, S]$ is not defined for $b > 0$, whereas for $b = 0$ and for each subset $S \subseteq \texttt{path}(d_{j-N})$ (of size $\leq \min\{B, \log N + 1\}$) we define

$$M[j, 0, S] = \frac{|d_{j-N} - \sum_{c_k \in S} \text{sign}_{j-N,k} \cdot c_k|}{r},$$

where $r = \max\{|d_{j-N}|, \mathsf{s}\}$ for $\text{err} = \text{relErr}$, and $r = 1$ for $\text{err} = \text{absErr}$.

In the case of an *internal error-tree node* c_j with $j < N$, our DP algorithm has two distinct choices when computing $M[j, b, S]$, namely either drop coefficient c_j or keep it in the final synopsis. If we choose to *drop* c_j from the synopsis, then it is easy to see that the maximum error from c_j's two child subtrees (i.e., c_{2j} and c_{2j+1}) will be propagated upward; thus, the minimum possible maximum error $M[j, b, S]$ for T_j in this case is simply

$$\min_{0 \leq b' \leq b} \max\left\{ M[2j, b', S] \ , \ M[2j+1, b-b', S] \right\}. \qquad (2)$$

Note that, in the above recurrence, S contains proper ancestors of c_j which are clearly proper ancestors of c_{2j} and c_{2j+1} as well; thus, the right-hand side of the recurrence is well-defined as the DP computation proceeds bottom-up to reach c_j. If, on the other hand, we choose to *keep* c_j in the synopsis (assuming, of course, $b \geq 1$), the least possible error $M[j, b, S]$ for T_j is computed as

$$\min_{0 \leq b' \leq b-1} \max\left\{ M[2j, b', S \cup \{c_j\}] \ , \ M[2j+1, b-b'-1, S \cup \{c_j\}] \right\}. \qquad (3)$$

(Again, note that the right-hand side of the recurrence is well-defined.) The final value for $M[j, b, S]$ defined as the *minimum* of the two possible choices for coefficient c_j, i.e., Equations (2) and (3) above. A pseudo-code description of our optimal DP algorithm for deterministic maximum-error thresholding in one dimension (termed MinMaxErr) is depicted in Figure 3.

Time and Space Complexity. Given a node/coefficient c_j at level l of the error tree, our MinMaxErr algorithm considers at most $O(B)$ space allotments to the T_j subtree and at most $O(2^l)$ subsets of ancestors of c_j. Thus, the number of entries in our DP array $M[]$ that need to be computed for c_j is $O(B2^l)$. Furthermore, the time needed to compute each such entry is $O(\log B)$. To see this, note that for any fixed node k and ancestor subset S, $M[k, b', S]$ is a decreasing function of the space allotment b'. Thus, the optimal distribution point b' in Equations (2)–(3) (Steps 11–28 in MinMaxErr) can be computed using an $O(\log B)$-time *binary search* procedure, looking for the space allotment where the error values for the two child subtrees are equal or the adjacent pair of cross-over allotments. (To simplify the exposition, this binary-search procedure has been omitted from the pseudo-code description in Figure 3.) Clearly, the total number of error-tree nodes at level l is $O(2^l)$ making the

```
procedure MinMaxErr( $W_A$ , $B$ , root , $S$, err )
Input: Array $W_A = [c_0, \ldots, c_{N-1}]$ of $N$ Haar wavelet coefficients,
    space budget $B$ (number of retained coefficients), error-subtree
    root-node index root, subset of retained ancestors of root node
    $S \subseteq$ path(root), target maximum error metric err.
Output: Value of $M[$root$, B, S]$ according to our optimal dynamic
    program ($M[$root$, B, S]$.value), decision made for the root node
    ($M[$root$, B, S]$.retained), and space allotted to left child subtree
    ($M[$root$, B, S]$.leftAllot). (The last two are used for re-tracing
    the optimal solution to build the synopsis.)
begin
1.  if ($M[$root$, B, S]$.computed = true) then
2.    return $M[$root$, B, S]$.value   // optimal value already in $M[]$
3.  if ( $N \leq$ root $< 2N$ ) then        // leaf/data node
4.    if ( $B = 0$ ) then
5.      $M[$root$, B, S]$.value := $|d_{j-N} - \sum_{c_k \in S} \text{sign}_{j-N,k} \cdot c_k|$
6.      if ( err = relErr ) then
7.        $M[$root$, B, S]$.value := $\frac{M[\text{root}, B, S].\text{value}}{\max\{|d_{j-N}|, \mathsf{s}\}}$
8.    endif
9.  else
10.   $M[$root$, B, S]$.value := $\infty$
11.   for $b := 0$ to $B$ step 1 do            // first choice: drop root
12.     left := MinMaxErr( $W_A$ , $b$ , $2 *$ root , $S$, err )
13.     right := MinMaxErr( $W_A$ , $B - b$ , $2 *$ root $+ 1$ , $S$, err )
14.     if ( max{ left, right } $< M[$root$, B, S]$.value ) then
15.       $M[$root$, B, S]$.value := max{ left, right }
16.       $M[$root$, B, S]$.retained := false
17.       $M[$root$, B, S]$.leftAllot := $b$
18.     endif
19.   endfor
20.   for $b := 0$ to $B - 1$ step 1 do    // second choice: keep root
21.     left := MinMaxErr( $W_A$ , $b$ , $2 *$ root , $S \cup \{$root$\}$, err )
22.     right := MinMaxErr( $W_A$ , $B - b - 1$ , $2 *$ root $+ 1$ ,
                                         $S \cup \{$root$\}$, err )
23.     if ( max{ left, right } $< M[$root$, B, S]$.value ) then
24.       $M[$root$, B, S]$.value := max{ left, right }
25.       $M[$root$, B, S]$.retained := true
26.       $M[$root$, B, S]$.leftAllot := $b$
27.     endif
28.   endfor
29. endif
30. $M[$root$, B, S]$.computed := true
31. return ( $M[$root$, B, S]$.value )
end
```

Figure 3: The MinMaxErr Algorithm: Optimal Deterministic Thresholding for Maximum Error in One Dimension.

overall time complexity of our DP algorithm

$$O(\sum_{l=0}^{\log N} 2^l 2^l B \log B) = O(B \log B \sum_{l=0}^{\log N} 2^{2l}) = O(N^2 B \log B).$$

With respect to the space requirements of our scheme, note that, even though the overall size of our DP array $M[]$ is $O(N^2 B)$, our MinMaxErr algorithm does not actually require the entire array to be memory-resident at all times. For instance, the results for all descendants of a node c_j are no longer needed and can be swapped out of memory once the results for node c_j have been computed. With this small optimization, it is easy to see that our bottom-up DP computation never requires more than one active "line" of the $M[]$ array *per error-tree level*, where the size of such a line for a node at level l is $O(B2^l)$ (i.e., $O(B)$ space allotments and $O(2^l)$ ancestor subsets). Thus, the size of the memory-resident working set for our DP algorithm drops to only $O(\sum_{l=0}^{\log N} 2^l B) = O(NB)$. Our analysis for the one-dimensional case is summarized in the following theorem.

THEOREM 3.1. *Our* MinMaxErr *algorithm is an optimal deterministic thresholding scheme for building one-dimensional wavelet synopses that minimize the maximum relative error (or, maximum absolute error) in the data approximation.* MinMaxErr *runs in time $O(N^2 B \log B)$ and has a total-space (working-space) requirement of $O(N^2 B)$ (resp., $O(NB)$).* ∎

3.2 Multi-Dimensional Wavelet Thresholding

Our deterministic wavelet thresholding problem becomes significantly more complex for multi-dimensional wavelets, and directly extending our optimal one-dimensional DP formulation to the case of multiple dimensions fails to give a practical solution. Remember that, in the D-dimensional error-tree structure (Section 2.2), even though the tree depth remains $O(\log N)$, each node in the tree now contains up to $2^D - 1$ wavelet coefficients with the same support region and different quadrant signs (Figure 2). This implies that the total number of possible ancestor subsets S for a multi-dimensional coefficient at a level $l = \Theta(\log N)$ is $O(2^{\log N \cdot (2^D - 1)}) = O(N^{2^D - 1})$, rendering the exhaustive-enumeration DP scheme of Section 3.1 completely impractical, even for the relatively small data dimensionalities (i.e., $D =$ 2–5) where wavelet-based data reduction is typically employed. (It is well known that, due to the "dimensionality curse", wavelets and other space-partitioning schemes become ineffective above 5–6 dimensions [3, 5, 8, 11].)

In this section, we introduce two efficient, polynomial-time *approximation schemes* for deterministic multi-dimensional wavelet thresholding for maximum-error metrics. Both our approximation schemes are based on *approximate dynamic programs* that explore a much smaller number of options than the optimal DP formulation, while offering tunable ϵ-approximation guarantees for the final target maximum-error metric. More specifically, our first scheme can give ϵ-additive-error guarantees for maximum relative or absolute error, whereas our second scheme is a $(1+\epsilon)$-approximation algorithm for maximum absolute error.

3.2.1 An ϵ-Additive-Error Approximation Scheme for Absolute/Relative Error Minimization

Intuitively, our optimal one-dimensional DP scheme is based on exhaustively enumerating, for each error subtree, all the possible error values "entering" the subtree through the choices made on the path to the subtree's root node. The key technical idea in our first approximation scheme is to avoid this exhaustive enumeration and, instead, try to *approximately "cover"* the range of all possible error contributions for paths up to the root node of an error subtree using a much smaller number of (approximate) error values. Our approximation scheme is then going to be based on a much "sparser" DP formulation, that only tabulates this smaller set of error values.

Specifically, let R denote the maximum absolute coefficient value in the error tree, and let $\epsilon < 1$ denote the desired approximation factor. Clearly, the additive contribution to the absolute data-value reconstruction error from *any* possible path in the error tree is guaranteed to lie in the range $\mathcal{R} = [-R2^D \log N,\ +R2^D \log N]$. Our approximation scheme covers the entire $\mathcal{R}$ range using *error-value breakpoints* of the form $\pm(1+\epsilon)^k$, for a range of (contiguous) integer values for the exponent k. Note that the number of such breakpoints needed is essentially

$$O(\log_{1+\epsilon}(2R2^D \log N) \approx O\left(\frac{D + \log R + \log\log N}{\epsilon}\right),$$

for small values of $\epsilon < 1$; in other words, $k \in \mathcal{K} = \{0, 1, \ldots, O(\frac{D+\log R + \log\log N}{\epsilon})\}$. Now, let $\mathsf{round}_\epsilon(v)$ be a function that rounds any value $v \in \mathcal{R}$ *down to the closest value* in the set $\mathcal{E} = \{0\} \cup \{\pm(1+\epsilon)^k,\ k \in \mathcal{K}\}$; that is, letting $l = \log_{1+\epsilon}|v|$, we have $\mathsf{round}_\epsilon(v) = (1+\epsilon)^{\lfloor l \rfloor}$ if $v \geq 1$, $-(1+\epsilon)^{\lceil l \rceil}$ if $v \leq -1$, and 0 otherwise. The DP array $M^a[]$ for our approximation scheme tabulates the values $M^a[j,\ b,\ e]$ capturing the approximate maximum error (relative or absolute) in the T_j error subtree (rooted at node j), assuming a space budget of b coefficients allotted to T_j and an approximate/rounded additive error contribution of $e \in \mathcal{E}$ due to proper ancestors of node j being discarded from the synopsis.

The *base case* for the computation of $M^a[]$, i.e., the case of a leaf/data node $j \geq N$ in the error tree is fairly straightforward: once again, $M^a[j,\ b,\ e]$ is only defined for $b = 0$, and $M^a[j, b, e] = \frac{|e|}{r}$, where r is either $\max\{|d_{j-N}|, \mathsf{s}\}$ or 1 depending on whether we are targeting a relative or absolute error metric.

In the case of an *internal error-tree node* j, remember that each node now corresponds to a *set* $S(j)$ of at most $2^D - 1$ (non-zero) coefficients, and has at most 2^D child subtrees (with indices, say, $j_1, \ldots, j_m$). Assume that we choose to maintain a subset $s \subseteq S(j)$ of node j's coefficients in the synopsis and, for each coefficient $c \in S(j)$, let $\mathsf{sign}(c, j_i)$ denote the sign of c's contribution to the j_i child subtree; then, we can estimate the least possible maximum error entries $M^a[j, b, e]$, $e \in \mathcal{E}$, for T_j (assuming, of course, that $b \geq |s|$) as

$$\min_{0 \leq b_1 + \ldots + b_m \leq b - |s|} \ \max_{1 \leq i \leq m} \{ \\ M^a[j_i, b_i, \mathsf{round}_\epsilon(e + \sum_{c \in S(j) - s} \mathsf{sign}(c, j_i) \cdot c)] \ \}.$$

In other words, for a given selected coefficient subset s, we consider all possible allotments of the remaining $b - |s|$ space to children of node j, with the rounded cumulative error that enters those children taking into account the contribution from the dropped coefficients in $S(j) - s$. To avoid the $O(B^{2^D})$ factor in run-time complexity implied by the search of all possible space allotments $b_1, \ldots, b_m$ to child subtrees in the above recurrence, we can simply *order* the search among a node's children (in a manner similar to [8]). The basic idea is to generalize our approximate DP-array entries to $M^a[\mathbf{j}, b, \mathbf{e}]$, where $\mathbf{j} = (j_1, \ldots, j_k)$ is a *list* of error-tree nodes and $\mathbf{e} = (e_1, \ldots, e_k)$ is a *list* of "incoming" additive errors corresponding to the nodes in $\mathbf{j}$. The above recurrence for $M^a[(j), b, (e)]$ then becomes simply

$$\min_{0 \leq b' \leq b - |s|} \ \max\{ \ M^a[(j_1), b', (e_1)] \ , \\ M^a[(j_2, \ldots, j_m), b - b' - |s|, (e_2, \ldots, e_m)] \ \},$$

where $e_k = \mathsf{round}_\epsilon(e + \sum_{c \in S(j) - s} \mathsf{sign}(c, j_k) \cdot c)$, for $k = 1, \ldots, m$. This generalization comes at a moderate increase of $O(2^D)$ in terms of time and space complexity over the one-dimensional case, and allows for an $O(\log B)$-time search for the breakup of a node's allotment to its children (using binary search in the above recurrence, as described earlier in the paper). The final approximate error value $M^a[(j), b, (e)]$ is computed as the *minimum* over all possible choices for the subset $s \subseteq S(j)$ of retained coefficients at node j, giving an overall time complexity of $O(2^{2^D-1}\log B)$ for computing the $M^a[(j),\ b,\ (e)]$ entry. (Again, remember that D is a small constant, typically between 2–5.) The following theorem summarizes the results of our analysis for our approximate deterministic-thresholding scheme.

THEOREM 3.2. *The above-described approximation scheme for deterministic multi-dimensional wavelet thresholding discovers an approximate solution that is guaranteed to be within a worst-case additive error of ϵR (resp., $\epsilon\frac{R}{\mathbf{S}}$) of the optimal (i.e., minimum) maximum absolute (resp., relative) error in time $O(\frac{D+\log R+\log\log N}{\epsilon}\ 2^{2^D+2D}N\log N\ \ B\log B)$ and with a total space requirement of $O(\frac{D+\log R+\log\log N}{\epsilon}$ $2^{2D}N\log NB)$.* ∎

Proof (sketch): Clearly, the error in our technique comes from the repeated rounding of incoming additive-error values at different nodes of the error tree; thus, the key question is how bad this incoming-error approximation can become due to repeated rounding. Fix a specific error-tree node j, and let e_t and s_t denote the true incoming error value and selected subset of coefficients (respectively) at node j in the optimal solution. Clearly, from our discussion above, the search space of our approximation scheme is going to include the optimal space allotments to each child subtree j_k of j, assuming a rounded incoming error value of

$$\mathsf{round}_\epsilon(\mathsf{round}_\epsilon(e_t) + \sum_{c\in S(j)-s_t}\mathsf{sign}(c, j_k)\cdot c),$$

which can be easily shown to be in the range

$$(e_t + \sum_{c\in S(j)-s_t}\mathsf{sign}(c, j_k)\cdot c) \pm \epsilon\cdot(|e_t| + \sum_{c\in S(j)-s_t}|c|).$$

(Remember that the values of e_t and c may very well be negative.) Thus, based on the properties of the multi-dimensional error-tree structure, it is easy to see that the *worst-case* additive deviation from the true absolute-error value is going to be at most $\epsilon R 2^D\log N$. Setting $\epsilon' = \frac{\epsilon}{2^D\log N}$, gives a worst-case additive deviation of ϵR (resp., $\epsilon\frac{R}{\mathbf{S}}$) from the optimal absolute (resp., relative) error. The running-time and space complexity of our approximation scheme for the above value of ϵ' follow immediately from our earlier discussion. ∎

We should stress here that the bounds in Theorem 3.2 represent a truly pathological worst-case scenario for our scheme, where all coefficients on a root-to-leaf path are of maximum absolute value R. In real-life applications, most of the energy of a data signal (i.e., array) is typically concentrated in a few large wavelet coefficients [3, 10, 15, 21], which implies that most coefficient values in the error tree will be (much) smaller than R. Thus, for real-life practical scenarios, we would typically expect our approximation scheme to be *much closer* to the optimal solution than the worst-case deviation bounds asserted in Theorem 3.2.

3.2.2 A $(1+\epsilon)$ Approximation Scheme for Absolute Error Minimization

We now consider the special case of minimizing absolute error for deterministic mult-dimensional wavelet thresholding, and propose a novel, polynomial-time $(1+\epsilon)$-approximation scheme. Our discussion here assumes that all wavelet coefficients are *integers* – we can always satisfy this assumption by appropriately *scaling* the coefficient values. For example, for integer data values d_i (e.g., a multi-dimensional frequency count array), scaling by a factor of $O(2^{D\log N}) = O(N^D)$ is always guaranteed to give integer Haar coefficients. Let R_Z denote the maximum (scaled) coefficient value in the error tree; as previously, it is easy to see that the additive (integer) contribution to the absolute reconstruction error from any possible path in the Haar error tree is guaranteed to lie in the integer range $\mathcal{R_Z} = [-R_Z 2^D\log N, +R_Z 2^D\log N]$. This observation directly leads to an *optimal pseudo-polynomial time algorithm* for our maximum absolute-error minimization problem. (In fact, this pseudo-polynomial time scheme directly extends to maximum relative-error minimization as well.) The key idea of our $(1+\epsilon)$-approximation scheme for absolute error is then to intelligently *scale-down* the coefficients in the error tree so that the possible range of integer additive-error values entering a subtree is *polynomially-bounded*.

We start by describing our optimal pseudo-polynomial time scheme. Briefly, our scheme is again based on dynamic-programming over the error tree of integer coefficient values, and follows along similar lines as our additive-error approximation algorithm presented in Section 3.2.1, but without doing any "rounding" of incoming error values. Briefly, our DP scheme constructs a table $M[j, b, e]$ for every node j, space budget $b \le B$ and error value e, $e \in \mathcal{R_Z}$, where $M[j, b, e]$ denotes the minimum possible maximum absolute error in the T_j subtree, assuming a space budget of b coefficients in T_j and an error contribution of e due to ancestors of node j. As earlier, we define $M[j, 0, e] = |e|$ for a leaf/data node j, whereas for an internal node j (with children $j_1, \ldots, j_m$) and assuming that a subset $s \subseteq S(j)$ of coefficients is retained in the synopsis, we compute $M[j, b, e]$ as

$$\min_{0\le b_1+\ldots+b_m\le b-|s|}\ \max_{1\le i\le m}\{\ M[j_i, b_i, e + \sum_{c\in S(j)-s}\mathsf{sign}(c, j_i)\cdot c]\ \},$$

where $\mathsf{sign}(c, j_i)$ is the sign of c's contribution to the j_i child subtree (as in Section 3.2.1). Once again, the $O(B^{2^D})$ factor needed to cycle through all the $b_1, \ldots, b_m$ allotments can be avoided using the generalization described earlier; thus, we have an optimal DP algorithm that runs in time $O(R_Z$ $2^{2^D+2D}\ N\log N\ \ B\log B)$. The key observation here is that, if R_Z is *polynomially bounded*, then the above-described DP scheme is also a polynomial-time algorithm. We use this idea to devise a polynomial-time $(1+\epsilon)$-approximation scheme.

Given a threshold parameter $\tau > 0$, we define a *truncated DP algorithm* as follows. Let $S_{>\tau}$ denote the set of coefficients with absolute value greater than τ, i.e., $S_{>\tau} = \{c \in \mathtt{coeff}(T_0) : |c| > \tau\}$, and define K_τ as the quantity $K_\tau = \frac{\epsilon\tau}{2^D\log N}$. Our truncated DP algorithm replaces each coefficient c in the error tree with a *scaled-down* coefficient value $c^\tau = \lfloor\frac{c}{K_\tau}\rfloor$, and works with these scaled (integer) coefficients; furthermore, our algorithm *always retains* all coefficients in $S_{>\tau}$ in the synopsis. More formally, we build a DP array $M^\tau[j, b, e]$ using the scaled coefficient values as follows. As previously, for a leaf node j, we define $M^\tau[j, 0, e] = |e|$. For an internal node j, our algorithm cycles through only those subsets $s \subseteq S(j)$ such that *s contains all coefficients in $S(j)\cap S_{>\tau}$* (the $M^\tau[j, b, e]$ entry is undefined if $b < |S(j)\cap S_{>\tau}|$). The DP recurrence for computing

$M^\tau[j, b, e]$ is identical to the one for our pseudo-polynomial scheme above, except for the fact that c is replaced by its scaled version c^τ.

We claim that the above truncated dynamic program is a *polynomial-time algorithm* for any value of the threshold parameter τ. Indeed, since we always retain all coefficients in $S_{>\tau}$, a coefficient c is dropped from the synopsis only if its scaled version c^τ satisfies $|c^\tau| \leq 2^D \log N/\epsilon$. This, of course, implies that the absolute additive error that can enter any subtree in our truncated DP algorithm is at most $2^{2D} \log^2 N/\epsilon$; in other words, the range of possible (integer) incoming error values e for our truncated DP array $M^\tau[]$ is guaranteed to be only $\mathcal{R_Z}^\tau = [-\frac{1}{\epsilon}2^{2D} \log^2 N, +\frac{1}{\epsilon}2^{2D} \log^2 N]$. Thus, based on our earlier analysis, the running time of our truncated DP algorithm for a fixed parameter τ is only $O(\frac{1}{\epsilon}2^{2^D+3D}\ N \log^2 N\ B \log B)$.

Given a threshold parameter τ, our truncated DP algorithm selects a subset $\mathcal{C}_\tau$ of coefficients to retain in the synopsis (note that this set is not defined if $B < |S_{>\tau}|$). Our absolute-error approximation scheme employs the truncated DP algorithm for each value $\tau \in \{2^k : k = 0, \ldots, \lceil \log R_Z \rceil\}$, and finally selects the synopsis $\mathcal{C}_\tau$ that minimizes the maximum absolute error in the data-value reconstruction. Clearly, since we only try $O(\log R_Z)$ different values for τ, the running time of our approximation algorithm remains polynomial.

We now demonstrate that the above-described scheme gives a $(1 + \epsilon)$-approximation algorithm for maximum absolute error minimization. Consider the optimal maximum absolute error synopsis $\mathcal{C}_{\rm OPT}$, and let $\mathsf{absErr}(\mathcal{C}_{\rm OPT})$ denote the corresponding maximum absolute error value. Also, let C denote the *maximum absolute coefficient value not retained in* $\mathcal{C}_{\rm OPT}$. Clearly, our approximation algorithm is going to try a threshold parameter, say τ', such that $\tau' \in [C, 2C)$. Our goal is to show that the maximum absolute error achieved by $\mathcal{C}_{\tau'}$ (i.e., $\mathsf{absErr}(\mathcal{C}_{\tau'})$) is very close to that achieved by the optimal solution $\mathcal{C}_{\rm OPT}$.

First, note that, by the definition of C and τ', the optimal solution is also guaranteed to retain all coefficients greater that τ', i.e., $S_{>\tau'} \subseteq \mathcal{C}_{\rm OPT}$. Thus, $\mathcal{C}_{\rm OPT}$ is obviously a feasible solution to our truncated DP instance (with threshold $= \tau'$). Now, let $\mathsf{absErr}_{\tau'}(\mathcal{C}_{\tau'})$, $\mathsf{absErr}_{\tau'}(\mathcal{C}_{\rm OPT})$ denote the maximum absolute errors in the $K_{\tau'}$*-scaled instance* for the $\mathcal{C}_{\tau'}$ synopsis (obtained by our truncated DP scheme) and the optimal $\mathcal{C}_{\rm OPT}$ synopsis, respectively. Given the optimality of our truncated dynamic program for the scaled instance, clearly

$$\mathsf{absErr}_{\tau'}(\mathcal{C}_{\tau'}) \leq \mathsf{absErr}_{\tau'}(\mathcal{C}_{\rm OPT}). \quad (4)$$

Let $\mathcal{C}$ be any subset of Haar coefficients. Obviously, in a $K_{\tau'}$-scaled instance, any coefficient $c \in \mathcal{C}$ is represented by $c^{\tau'} = \lfloor \frac{c}{K_{\tau'}} \rfloor$ which differs by at most 1 from $\frac{c}{K_{\tau'}}$; thus, it is easy to see that the scaled and non-scaled maximum absolute errors for $\mathcal{C}$ are related as follows

$$\mathsf{absErr}(\mathcal{C}) \in \left(K_{\tau'} \mathsf{absErr}_{\tau'}(\mathcal{C}) \pm K_{\tau'} 2^D \log N \right). \quad (5)$$

Applying the above formula for $\mathcal{C} = \mathcal{C}_{\rm OPT}$ and combining with Equation (4), we have

$$\begin{aligned} \mathsf{absErr}(\mathcal{C}_{\rm OPT}) &\geq K_{\tau'} \mathsf{absErr}_{\tau'}(\mathcal{C}_{\rm OPT}) - K_{\tau'} 2^D \log N \\ &\geq K_{\tau'} \mathsf{absErr}_{\tau'}(\mathcal{C}_{\tau'}) - K_{\tau'} 2^D \log N, \end{aligned}$$

and using Equation (5) once again with $\mathcal{C} = \mathcal{C}_{\tau'}$, we get

$$\mathsf{absErr}(\mathcal{C}_{\tau'}) \leq K_{\tau'} \mathsf{absErr}_{\tau'}(\mathcal{C}_{\tau'}) + K_{\tau'} 2^D \log N.$$

Now, simply combining the last two formulas and substituting $K_{\tau'} = \frac{\epsilon\tau'}{2^D \log N}$, we have

$$\begin{aligned} \mathsf{absErr}(\mathcal{C}_{\tau'}) &\leq \mathsf{absErr}(\mathcal{C}_{\rm OPT}) + 2K_{\tau'} 2^D \log N \\ &\leq \mathsf{absErr}(\mathcal{C}_{\rm OPT}) + 2\epsilon\tau'. \end{aligned} \quad (6)$$

PROPOSITION 3.3. *Let $\mathcal{C}_{OPT}$ and τ' be as defined above. Then,* $\mathsf{absErr}(\mathcal{C}_{OPT}) > \frac{\tau'}{2}$. ∎

Proof (sketch): Our key claim here is that if an absolute coefficient value C is dropped from the optimal synopsis $\mathcal{C}_{\rm OPT}$, then there exists some data value d_i in the underlying data domain such that the absolute error in the reconstruction of d_i is at least C; that is, $\mathsf{absErr}_i \geq C$. To see this, observe that for any coefficient c in the Haar error tree, there exist non-empty subsets of child subtrees where c contributes with either possible sign (i.e., a "+" subset as well as a "-" subset). Thus, given a dropped coefficient with an absolute value of C, we can always appropriately "navigate" the signs through the error-tree structure to discover a path from the coefficient to a leaf/data node d_i where the absolute error contribution of the coefficient is C (regardless of whether other coefficients in the path have also been dropped). The detailed argument can be found in the full paper [9]. Thus, $\mathsf{absErr}(\mathcal{C}_{\rm OPT}) \geq C$. But, by our choice of τ', we have $\tau' \in [C, 2C)$, or $C > \frac{\tau'}{2}$. The result follows. ∎

Combining Inequality (6) with Proposition 3.3, we have

$$\mathsf{absErr}(\mathcal{C}_{\tau'})\ \leq\ (1 + 4\epsilon)\ \mathsf{absErr}(\mathcal{C}_{\rm OPT}).$$

Thus, simply setting $\epsilon' = \epsilon/4$, we have a $(1 + \epsilon)$-approximation scheme for maximum absolute error minization in multiple dimensions. The following theorem summarizes our analysis.

THEOREM 3.4. *The above-described approximation scheme for deterministic multi-dimensional wavelet thresholding discovers an approximate solution that is guaranteed to be within $(1+\epsilon)$ of the optimal (i.e., minimum) maximum absolute error in time $O(\frac{\log R_Z}{\epsilon} 2^{2^D+3D}\ N \log^2 N\ B \log B)$ and with a total space requirement of $O(\frac{1}{\epsilon}2^{3D}\ N \log^2 N\ B)$.* ∎

4. RELATED WORK

Wavelets have a long history of successes in the signal and image processing arena [14, 19, 20] and, recently, they have also found their way into data-management applications. Matias et al. [15] first proposed the use of Haar-wavelet coefficients as synopses for accurately estimating the selectivities of range queries. Vitter and Wang [21] describe I/O-efficient algorithms for building multi-dimensional Haar wavelets from large relational data sets and show that a small set of wavelet coefficients can efficiently provide accurate approximate answers to range aggregates over OLAP cubes. Chakrabarti et al. [3] 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. [16] 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. [10] propose algorithms for building approximate one-dimensional Haar-wavelet synopses over numeric data streams. Deligiannakis and Roussopoulos [4] introduce time- and space-efficient techniques for constructing Haar-wavelet synopses for data sets with multiple measures (such as those typically found in OLAP applications).

All the above papers rely on conventional, L_2-error-based thresholding schemes that typically decide the significance of a coefficient based on its absolute normalized value. Garofalakis and Gibbons [7, 8] have shown that such conventional wavelet synopses can suffer from several important problems, including the introduction of severe bias in the data reconstruction and wide variance in the quality of the data approximation, as well as the lack of non-trivial guarantees for individual approximate answers. In contrast, their proposed *probabilistic wavelet synopses* rely on a a probabilistic thresholding process based on *randomized rounding* [17], that tries to *probabilistically* control the maximum relative error in the synopsis by minizing appropriate probabilistic metrics (like, normalized standard error or normalized bias). The problem addressed in this paper, namely the design of efficient *deterministic* thresholding schemes for maximum error metrics, is one of the main open problems posed by their study [8].

There is a rich mathematics literature on m-term approximations using wavelets (m is the number of coefficients in the synopsis). Some prior work has studied thresholding approaches for meeting a target upper bound for an L_p-error metric [6, 20]. We are not aware of work addressing the deterministic minimization of relative errors with sanity bounds (arguably the most important scenario for approximate query processing in databases) and, to the best of our knowledge, ours are the first results on computationally-efficient (optimal and near-optimal) deterministic thresholding schemes for minimizing maximum-error metrics for one- and multi-dimensional wavelet summaries.

5. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this paper, we have proposed novel, computationally-efficient schemes for deterministic maximum-error wavelet thresholding in one and multiple dimensions. For one-dimensional wavelets, we have introduced an optimal, low polynomial-time thresholding algorithm based on a new Dynamic-Programming formulation that can be used to minimize either the maximum relative error or the maximum absolute error in the data approximation. For the multi-dimensional case, we have designed novel, polynomial-time approximation schemes (with tunable ϵ-approximation guarantees for the target metric) for maximum-error thresholding based on approximate dynamic programs.

There are several interesting directions for future research in this area. As demonstrated in this paper, deterministic Haar-wavelet thresholding for maximum-error metrics becomes significantly more difficult as the data dimensionality increases (similar observations have also been made for the related problem of *histogram construction* [18]); however, a formal $\mathcal{NP}$-hardness proof for our multi-dimensional wavelet thresholding problem still eludes our efforts. The question of designing an efficient $(1+\epsilon)$-approximation scheme for maximum relative error in multiple dimensions is also left open. Another interesting direction involves studying the relative average-case performance of the schemes presented in this paper and the probabilistic synopses of [7, 8] over real-life data sets; we are currently implementing our techniques and hope to report our experimental findings in the near future. Finally, an important question in this realm concerns the general 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 bases that are better suited for optimizing, for example, relative-error metrics in the data approximation?

6. REFERENCES

[1] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. "Join Synopses for Approximate Query Answering". In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 275–286, Philadelphia, Pennsylvania, May 1999.

[2] Laurent Amsaleg, Philippe Bonnet, Michael J. Franklin, Anthony Tomasic, and Tolga Urhan. "Improving Responsiveness for Wide-Area Data Access". *IEEE Data Engineering Bulletin*, 20(3):3–11, September 1997. (Special Issue on Improving Query Responsiveness).

[3] Kaushik Chakrabarti, Minos Garofalakis, Rajeev Rastogi, and Kyuseok Shim. "Approximate Query Processing Using Wavelets". In *Proceedings of the 26th International Conference on Very Large Data Bases*, pages 111–122, Cairo, Egypt, September 2000.

[4] Antonios Deligiannakis and Nick Roussopoulos. "Extended Wavelets for Multiple Measures". In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, San Diego, California, June 2003.

[5] Amol Deshpande, Minos Garofalakis, and Rajeev Rastogi. "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, California, May 2001.

[6] R. A. DeVore. "Nonlinear Approximation". *Acta Numerica*, 7:51–150, 1998.

[7] Minos Garofalakis and Phillip B. Gibbons. "Wavelet Synopses with Error Guarantees". In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 476–487, Madison, Wisconsin, June 2002.

[8] Minos Garofalakis and Phillip B. Gibbons. "Probabilistic Wavelet Synopses". *ACM Transactions on Database Systems*, 29(1), March 2004. (SIGMOD/PODS Special Issue).

[9] Minos Garofalakis and Amit Kumar. "Deterministic Wavelet Thresholding for Maximum-Error Metrics". Bell Labs Technical Memorandum, December 2003.

[10] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin J. Strauss. "Surfing Wavelets on Streams: One-pass Summaries for Approximate Aggregate Queries". In *Proceedings of the 27th International Conference on Very Large Data Bases*, Roma, Italy,

September 2001.

[11] Dimitrios Gunopulos, George Kollios, Vassilis J. Tsotras, and Carlotta Domeniconi. “Approximating Multi-Dimensional Aggregate Range Queries Over Real Attributes”. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, Texas, May 2000.

[12] Peter J. Haas and Arun N. Swami. “Sequential Sampling Procedures for Query Size Estimation”. In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*, pages 341–350, San Diego, California, June 1992.

[13] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. “Online Aggregation”. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, May 1997.

[14] Björn Jawerth and Wim Sweldens. “An Overview of Wavelet Based Multiresolution Analyses”. *SIAM Review*, 36(3):377–412, 1994.

[15] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. “Wavelet-Based Histograms for Selectivity Estimation”. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 448–459, Seattle, Washington, June 1998.

[16] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. “Dynamic Maintenance of Wavelet-Based Histograms”. In *Proceedings of the 26th International Conference on Very Large Data Bases*, Cairo, Egypt, September 2000.

[17] Rajeev Motwani and Prabhakar Raghavan. *“Randomized Algorithms”*. Cambridge University Press, 1995.

[18] S. Muthukrishnan, Viswanath Poosala, and Torsten Suel. “On Rectangular Partitionings in Two Dimensions: Algorithms, Complexity, and Applications”. In *Proceedings of the Seventh International Conference on Database Theory (ICDT’99)*, Jerusalem, Israel, January 1999.

[19] Apostol Natsev, Rajeev Rastogi, and Kyuseok Shim. “WALRUS: A Similarity Retrieval Algorithm for Image Databases”. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, Philadelphia, Pennsylvania, May 1999.

[20] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. *“Wavelets for Computer Graphics – Theory and Applications”*. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[21] Jeffrey Scott Vitter and Min Wang. “Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets”. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, Philadelphia, Pennsylvania, May 1999.