Multiplicative Synopses for Relative-Error Metrics

Panagiotis Karras
School of Computing
National University of Singapore
lastname@comp.nus.edu.sg

ABSTRACT

Existing hierarchical summarization techniques fail to provide synopses good in terms of *relative*-error metrics. This paper introduces *multiplicative synopses*: a summarization paradigm tailored for effective relative-error summarization. This paradigm is inspired from previous hierarchical index-based summarization schemes, but goes beyond them by altering their underlying data representation mechanism. Existing schemes have decomposed the summarized data based on sums and differences of values, resulting in what we call *additive synopses*. We argue that the incapacity of these models to handle relative-error metrics stems exactly from this additive nature of their representation mechanism. We substitute this additive nature by a *multiplicative* one. We argue that this is more appropriate for achieving low-relative-error data approximations. We develop an efficient linear-time dynamic programming scheme for one-dimensional multiplicative synopsis construction under general relative-error-based metrics, and a special scheme for the case of *maximum* relative error. We generalize our schemes to higher data dimensionality and we show a surprising additional benefit gained by our special scheme for maximum relative error in this case. In our experimental study, we verify the higher efficacy of our model on relative-error-oriented summarization problems.

1. INTRODUCTION

A *data synopsis* aims to quickly and efficiently reduce a very large data set into a compact approximate representation that captures its basic features. Synopsis construction algorithms and related index structures provide a basic tool in a wide spectrum of applications, such as query optimization [24, 34], approximate query answering [42, 4], OLAP/DSS systems [50], time-series indexing [5], and distributed stream monitoring [46].

In all applications, the task is to construct a summary of the given data, stored within bounded space, while minimizing an appropriate error function. Given a *point query* that asks for a data value d_i, an estimate of d_i, say $\hat{d}_i$, is constructed and returned as an answer [22]. Previous research has tackled the problem in two principal ways: histogram-based techniques [24, 44, 43, 26, 42, 12, 13, 22, 40, 19, 48], and techniques based on hierarchical index structures such as that provided by the Haar wavelet transform [34, 50, 4, 8, 18, 10, 11, 36, 7, 16], the GenHist method [23], the compact hierarchical histogram [46] and the Haar$^+$ tree [28].

Many techniques achieve satisfactory results when the target error metric is a function of *absolute* point-wise errors [46, 28]. Still, error metrics of this class are not the most desirable ones; *relative error*-based metrics are arguably more important for most data approximation tasks [50, 10, 22, 11]. Given that individual values in a given data set may vary by orders of magnitude, it is rather desirable to approximate each of them within a good relative error, than to estimate all of them within a uniformly good absolute error, which may entail huge relative-error variations. For example, the value $x_i = 10$ estimated as $\hat{x}_i = 30$ entails the same absolute error as the value $x_j = 1010$ estimated as $\hat{x}_j = 990$, but the relative error is 200% in the former case but only 2% in the latter. Still, although much of related work, as [10, 11, 16], has considered relative-error metrics as an objective of optimization *among others*, only histogram construction [22] has been *specialized with regard to* this highly desirable class of error metrics. Thus, [22] treats the case of relative-error metrics sufficiently from the histograms' point of view. Such a *dedicated* treatment from the point of view of *hierarchical* summarization is pending. Unfortunately, the two most recently proposed hierarchical summarization techniques did not pay full attention to this important class of error metrics. The methods of [46] did not tailor their approximate values per bucket for such metrics (as [22] did), hence did not achieve good results with them; [28] did not examine such metrics in its evaluation. Besides, none of the recent hierarchical synopsis studies that included relative-error metrics in their evaluations [11, 16, 46] assessed its performance in comparison to [22]. Hence, research on hierarchical summarization has still not satisfied one of its central desiderata.

In this paper, we argue that the data reconstruction mechanism of a synopsis structure can be fruitfully specialized for the error metric at hand, contrary to the "one model fits all" approach of [10, 11, 16, 46, 28]. We introduce *multiplicative synopses*: a summarization model for relative-error-oriented problems. We develop linear-time algorithms for multiplicative synopsis construction, and a faster, *indirect* technique for *maximum* relative error. We generalize our model to higher dimensionality, showing that the benefit gained by

EDBT 2009, March 24–26, 2009, Saint Petersburg, Russia.

the indirect technique *grows* with it. Last, we experimentally verify the advantage of our approach over other methods.

2. BACKGROUND AND RELATED WORK

Given an n-sized data vector $\mathbf{D} = \langle d_0, d_1, \ldots, d_{n-1} \rangle$, the problem is to devise an n-length representation $\hat{\mathbf{D}}$ of $\mathbf{D}$ using at most B space units, so that a given *relative-error metric* in the approximation is minimized. A Minkowski-norm function:

$$\mathcal{L}_p^{\mathsf{rel}}\left(\hat{\mathbf{D}}, \mathbf{D}\right) = \left(\frac{1}{n}\sum_i \left(\frac{|\hat{d}_i - d_i|}{\max\{|d_i|, S\}}\right)^p\right)^{\frac{1}{p}} \quad (1)$$

covers most practically interesting point-wise relative-error functions; $\hat{d}_i$ is the reconstructed value for d_i; $S > 0$ is a sanity bound that prevents small values from unnaturally dominating the result [10, 11]. This sanity bound is set based on the needs of the application at hand; to be meaningful, it has to be larger than the smallest value in the data set. Past research on summarization can be divided in two broad categories: The former, *single-value summarization*, approximates the given data by dividing it into *buckets*. Typically these buckets contain consecutive values; alternatively, one bucket may contain another, forming a hierarchical structure, as in [3, 46]. An estimated data item is reconstructed as the value representing the bucket to which it belongs. The latter category, *additive summarization*, represents the data in terms of distinct elements that are *added* to each other, typically utilizing a hierarchical index structure. An estimated data item is reconstructed by *adding* an appropriate series of terms.

2.1 Single-value Summarization

2.1.1 Plain Histograms

The classical case of one-dimensional single-value summarization is a *plain histogram*; it divides the data vector $\mathbf{D}$ into $B \ll n$ disjoint intervals[1] $[b_i, e_i]$, $1 \leq i \leq B$ called *buckets* or *segments*, and attributes to each of them a single value v_i that approximates all consecutive values therein, d_j, $j \in [b_i, e_i]$. A single bucket (segment) can be expressed by the triplet $s_i = \{b_i, e_i, v_i\}$. Given a target metric, the optimal value of v_i is defined as a function [26, 48] of the data in $[b_i, e_i]$; such functions for several relative-error metrics are defined in [22]. An $O(n^2B)$-time dynamic-programming scheme that builds Euclidean-error-optimal histograms was offered in [26]. For an arbitrary error metric, it takes $O(n^3B)$ time. Its key observation is that the b-optimal histogram of $\mathbf{D}$ can be recursively derived from the space of $(b-1)$-optimal partitionings of its prefix vectors. A suite of efficient histogram construction algorithms for several relative-error-based metrics was delivered in [22]. Several works have proposed approximation and streaming algorithms for histogram construction [12, 13, 45, 17, 19, 48]; [25] suggested algorithms that identify the optimal *set* of histograms for a set of attributes under an expected workload, as opposed to independently optimizing the histogram for each attribute; other approaches have provided histogram algorithms for workload-based [39] and range-query [31, 14, 20, 38] optimization.

2.1.2 Compact Hierarchical Histograms

The Compact Hierarchical Histogram (CHH) [46] defines a binary hierarchy of intervals and selects a subset of nodes to represent a data set. A data item is approximated by the value of its lowest non-zero ancestor node in the CHH hierarchy (see Figure 7 that follows). Hence, the CHH is a case of single-value summarization. It can be easily shown that a binary CHH is equivalent to the *additive* hierarchical summarization structure proposed in [2]. Besides, it also forms a simplified variant of the more general (additive) Haar$^+$ model [28] (Section 2.2.2). [46] proposed exact solutions for limited versions of the CHH construction problem, and heuristics for the most general, longest-prefix-match CHH problem. The best-performing CHH algorithm is a greedy heuristic that improves upon an optimal *overlapping* partitioning; in such a partitioning, the value assigned at a CHH node is the optimal value for the whole data interval under its scope (as in a plain histogram bucket [26, 22, 48]), but not for the value set it *actually* approximates. The heuristic uses the occupied node positions computed for such an overlapping partitioning, but adjusts their values so as to fit the actually approximated data set (i.e., a subset of the data under the node's scope). Still, [46] did not customize its approach for the case of relative-error metrics, as [22] did with plain histograms.

2.1.3 Multidimensional Extensions

Several pieces of work have strived to extend the histogram idea to multiple dimensions, with a view at approximating the joint data distribution of a multi-attribute data set. [35] introduced a multidimensional version of the equi-depth histogram of [41]. [43] proposed MHist, a multidimensional histogram generalizing the MaxDiff heuristic of [44]. [1] presented STGrid, an algorithm that maintains multidimensional histograms by analyzing query results; [3] extended this work with STHoles, which allows a bucket to contain another; [47] provided ISOMER, a both consistent and efficient development of feedback-driven histogram maintenance. Still, heuristics based on query feedback are susceptible to errors for unseen data regions. Besides, even in the two-dimensional case, constructing an optimal histogram of arbitrary non-overlapping rectangular buckets is NP-hard [37]. Algorithms with approximation guarantees have been provided for limited versions of the problem, with non-arbitrary buckets [30, 40]. A suggestion for overcoming this difficulty is to identify the critical areas of dependence among dimensions in multidimensional data and capture them with a statistical interaction model, which can form the basis for lower-dimensional histograms to approximate the overall joint data distribution [9]. The technique of [49] maintains a sketch on the joint data distribution of a continuous stream, from which it can extract a multidimensional histogram on demand.

2.2 Additive Summarization

2.2.1 Transformation-based Additive Synopses

Another influential summarization technique compresses the data into a set of significant terms in a mathematical transformation (decomposition) [34, 32], and reconstructs them via the respective inverse transformation. The Discrete Cosine Transform (DCT) was proposed for that purpose in [32]. However, [32] chooses the set of DCT coefficients in the synopsis a priori, independently of the data, by means of static geometrical zonal sampling. This may not be the most appropriate set of coefficients, resulting in low-quality synopses. Still, in the Discrete Haar Wavelet Transform

[1] The bounds b_i and e_i are *indices*, not data values.

(DHWT) method, introduced in [34], synopsis terms are chosen with respect to the initial data set. The DHWT can be visualized by a complete binary tree, the *Haar tree*. The coefficient in the Haar tree root node contains the overall average value; each other coefficient value c_i adds $+c_i$ to data cells (leaves) in its *left* sub-tree and $-c_i$ to those in its *right* sub-tree. A data value is reconstructed by adding/subtracting the (signed) terms along a root-to-leaf path.

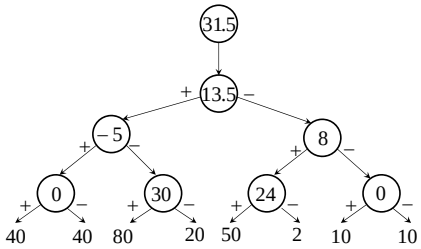


Figure 1: Full Haar Wavelet Decomposition

Figure 1 shows the Haar tree representing the complete DHWT for the vector $\mathbf{D} = \{40, 40, 80, 20, 50, 2, 10, 10\}$. Two out of eight obtained coefficients are 0, which is the neutral element of addition. Such *zero-valued coefficients* do not have to be stored; a data set is represented by the *non-zero terms*. A synopsis construction algorithm aims to specify an appropriate set of B non-zero terms to maintain. For example, coefficients -5 and 8 in Figure 1 can be set to 0, while the values of others may be adjusted so as to enhance the accuracy of approximation. Past research [10, 11, 16] has built upon [34], resulting, most recently, to the Haar$^+$ model.

2.2.2 The Haar$^+$ Model

The Haar$^+$ model [28] extended the model based on the DHWT by inserting extra coefficient nodes in the hierarchy; each of these coefficients contributes its (signed) value to a single dyadic interval alone. Therefore, the data reconstruction by a Haar$^+$ tree does not correspond to an inverse DHWT. Figure 2 depicts a simple one-dimensional Haar$^+$ tree that may approximate a four-element data set $\{d_0, d_1, d_2, d_3\}$; it contains a single root coefficient node c_0 that *adds* its value to all approximated data values, followed by a binary tree of triads (C_1, C_2 and C_3), which substitute the single non-root coefficients of the classical Haar tree. In each triad (e.g., C_1), the *head coefficient* (e.g., c_1) behaves as a classical Haar wavelet coefficient: it *adds* its (signed) value for the reconstruction of data cells in its left sub-tree and *subtracts* it for the reconstruction of those in its right sub-tree. The *supplementary coefficients* (e.g., c_2 and c_3) *add* their (signed) values only in the single subinterval that they affect (e.g., c_2 adds to d_0 and d_1). Haar$^+$ increases the accuracy of approximation in relation to the DHWT-based models and allows for faster synopsis construction as well [28].

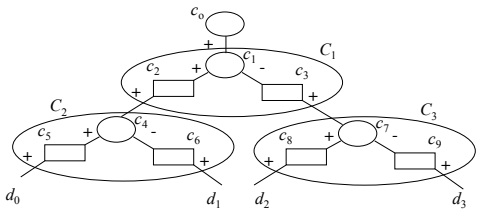


Figure 2: An One-Dimensional Haar$^+$ Tree

2.2.3 The GenHist Heuristic

[23] proposed GenHist, a *generalized* multidimensional histogram. GenHist defines multidimensional buckets of variable size, with unrestricted overlap among them. Such buckets are extracted from progressively coarser grids over the data set. The overlap among buckets allows for a more compact approximation of data distributions. Data regions that fall in the domain of more than one overlapping buckets are reconstructed by *adding* the contributions from all. Still, the GenHist heuristic, as it is designed for the hard multidimensional case, does not attempt to calculate optimal bucket values and/or positions for a given data set.

3. MOTIVATION

Both [10] and [11] concluded by questioning the "general suitability of the [then-used] Haar-wavelet transform as a data-summarization and approximate query processing tool", and inquired whether another model would be better suited for optimizing relative-error metrics in the data approximation. This paper presents our proposal for answering this question.

A data summarization mechanism suitable for relative-error metrics should naturally give more emphasis on achieving low absolute errors for *small* absolute values; hence, it can sacrifice the absolute approximation accuracy at data cells of larger absolute value for the sake of smaller ones. We argue that additive (and single-value) summarization is not suitable for that purpose. A more suitable approach would be to use *multiplication* as the basic data reconstruction mechanism. In the next section we proceed to do so by introducing the *multiplicative transform*.

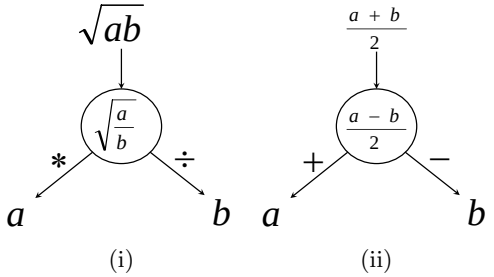


Figure 3: Multiplicative transform (i), and Haar wavelet transform (ii)

4. THE MULTIPLICATIVE TRANSFORM

The multiplicative transform analyzes a data set in hierarchical levels of detail based on pairwise products and ratios. Figure 3(i) depicts the basic decomposition step for a given pair of (positive) values $\{a, b\}$. The data are decomposed to the root of their product $\sqrt{ab}$ and the root of their ratio $\sqrt{\frac{a}{b}}$. They can be reconstructed by multiplying (dividing) the former by the latter. The figure depicts, along arrows, the operations that have to be performed between the *incoming value* to a decomposition node and the *stored coefficient* in it, in order to *reconstruct* the data. For comparison, the basic step of a Haar wavelet transform [34] is depicted in Figure 3(ii). In this case, the data are decomposed to their average $\frac{a+b}{2}$ and their semi-difference $\frac{a-b}{2}$, and can be reconstructed by adding (subtracting) the latter to (from) the former. The operators of *multiplication*, *division* and *root* now play the role that *addition*, *subtraction* and *average* play in the Haar wavelet transform. Hence, in fact, the

multiplicative transform is equivalent to an additive transform in the *logarithmic* domain.
An example of a complete *multiplicative decomposition* for the data vector $\mathbf{D} = \{40, 40, 80, 20, 50, 2, 10, 10\}$ is depicted in Figure 4. Four out of eight obtained coefficients are equal to 1, which is the neutral element of multiplication. Such *unit coefficients* do not have to be stored. Hence, an eight-element data set has been compressed to a four-term decomposition. Further compression is possible by setting further *non-unit coefficients* to the value 1, while adjusting the values of retained non-unit coefficients so as to obtain the highest possible accuracy in data approximation.

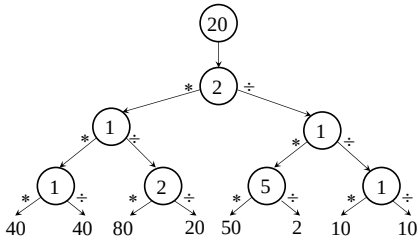


Figure 4: Full Multiplicative Decomposition

We argue that the multiplicative transform is naturally well-suited for relative-error-based metrics. We justify our argument with the following theorem, which shows that the multiplicative transform behaves for relative-error-metrics as the DHWT behaves for absolute-error-based metrics.

THEOREM 1. *Let z be a Haar wavelet coefficient that needs to approximate a data pair $\{d_i, d_j\}$ with incoming value v. Then, for any v and any Minkowski-norm absolute-error-based metric $\mathcal{L}_p$, the optimal value to be used is the DHWT semi-difference itself, $z = \frac{d_i - d_j}{2}$. Similarly, if z is a multiplicative transform coefficient approximating $\{d_i, d_j\}$ with incoming value v, then for any v and any Minkowski-norm relative-error-based metric $\mathcal{L}_p^{\mathsf{rel}}$, the optimal value to be used is the multiplicative transform root-of-ratio itself, $z = \sqrt{\frac{d_i}{d_j}}$.*

PROOF. In both cases, the value z equalizes the errors at d_i and d_j and hence minimizes the error function. In the DHWT case, the absolute errors $\left|v + z - d_i\right|$ and $\left|v - z - d_j\right|$ are equalized at $\left|v - \frac{d_i + d_j}{2}\right|$. In the multiplicative case, the relative errors $\left|\frac{vz}{d_i} - 1\right|$ and $\left|\frac{v}{zd_j} - 1\right|$ are equalized at $\left|\frac{v}{\sqrt{d_i d_j}} - 1\right|$. □

Having introduced the multiplicative transform, we can combine it with any synopsis construction algorithm developed for its additive counterpart, the Haar wavelet transform. Thus, we can build a dynamic-programming algorithm that identifies an optimal subset of coefficients to retain for a given error metric; this approach would result into *restricted* multiplicative synopses, by analogy to the restricted Haar wavelet synopses [11]. Besides, we could also create multiplicative variants of workload-oriented restricted Haar wavelet synopsis algorithms such as those of [36, 33, 6, 21]. However, all these models have been outdated by the *unrestricted* model [16], which effectively increases the accuracy in synopsis construction both in the general and the workload-oriented case (a relative-error-based problem can be seen as a case of workload-oriented problem). Thus, we can design an *unrestricted* multiplicative synopsis algorithm that identifies appropriate coefficient positions and values for a given error metric (and workload). Still, the unrestricted model itself has been surpassed by the Haar$^+$ model [28], which improves it both on accuracy of approximation and algorithmic complexity. Thus, we are in a position to directly offer a state-of-the-art multiplicative synopsis algorithm, inspired from [28], without considering the multiplicative analogues of [11, 36, 33, 6, 21, 16]. The next section proceeds to do so by introducing the *multiplicative synopsis tree*.

5. THE MS-TREE

Figure 5 depicts a simple one-dimensional Multiplicative Synopsis tree (MS-Tree) that may summarize a four-element data set $\{a, b, c, d\}$. The root coefficient node c_0 is a factor of all reconstructed data values. A binary tree of coefficient nodes follows. These coefficients are grouped in triplets. Each triplet C includes a *main coefficient*, such as c_1, c_2, and c_3, which behaves as a multiplicative transform coefficient: it is a factor of data values in the interval of its left subtree, and an inverse factor of data values in the interval of its right subtree. Furthermore, each main coefficient is augmented by two (left and right) *escort coefficients*, such as c_{1L} and c_{1R}, c_{2L} and c_{2R}, c_{3L} and c_{3R}; each of them is a factor of the data values under its scope. For example, c_2 is a factor of a, while c_2^{-1} is a factor of b; c_{1R} is a factor of both c and d.

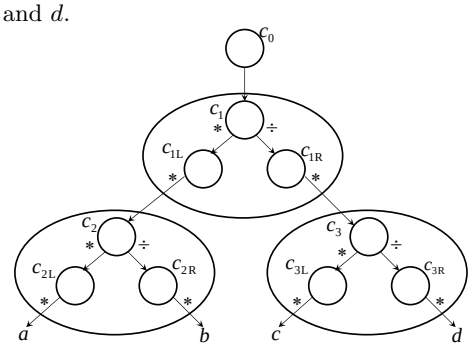


Figure 5: An MS-Tree

An *optimal* synopsis of space budget B for a given error metric $\mathcal{E}$ places B (positive) non-unit coefficient values at *any* positions in the MS-Tree tree so that $\mathcal{E}$ is minimized; unoccupied coefficients implicitly assume the neutral value 1. For example, a 2-term MS-Tree synopsis for the four-element data set $\{50, 2, 9, 11\}$ consists of the coefficients $\{c_0 = 10, c_2 = 5\}$ and produces the approximation $\{50, 2, 10, 10\}$ with relative-error values $\{0, 0, \frac{1}{9}, \frac{1}{11}\}$. Such quality of approximation is not achievable with existing single-value and additive summarization techniques. For example, a 2-bucket histogram for the same data set would approximate it as $\{50, 2, 2, 2\}$ with relative-error values $\{0, 0, \frac{7}{9}, \frac{9}{11}\}$. This example shows the power of the MS-Tree to achieve good approximation in relative-error terms. By default, an MS-Tree approximates absolute magnitudes. Data sets including *negative* values can be accommodated by storing a bitmap of signs and summarizing magnitudes. In fact, most real-world applications (e.g., selectivity estimation, stock exchange indices, internet traffic monitoring, etc.) concern data of positive value. Besides, in its default form, an MS-Tree approximates data vectors of *binary* (i.e., power of 2) length. Other lengths can be accommodated by filling in (padding) the missing parts with unit values, as with DHWT methods [34, 27].
We introduce some convenient notation for the discussion that follows: $a \in \mathsf{path}(b)$ denotes that node a lies on the

path from the root of the tree to leaf node b. A data item d_j of the represented data vector $\mathbf{D}$ is reconstructed as $d_j = \prod_{c_i \in \mathsf{path}(d_j)} c_i^{\delta_{ij}} c_{iD_{ij}}$, where c_i is a main coefficient and, if d_j lies in its left subtree, $\delta_{ij} = +1$ and $D_{ij} = L$, otherwise $\delta_{ij} = -1$ and $D_{ij} = R$. The *state* of a given triplet $C_i = \{c_i, c_{iL}, c_{iR}\}$ is a four-element vector $[v, a, b, c]$, where v is the *incoming value* at c_i, that is, the value reconstructed from the root of the MS-Tree to the node of c_i, $v = \prod_{c_k \in \mathsf{path}(c_i)} c_k^{\delta_{ki}} c_{kD_{ki}}$, while a, b, c are respectively the values at c_i, c_{iL} and c_{iR}. A state of a triplet C_i *creates* the *outgoing pair* $\left[vba, \frac{vc}{a}\right]$; that is, the incoming value to the child of c_{iL} (node c_{2i}) is vba, while that to the child of c_{iR} (node c_{2i+1}) is $\frac{vc}{a}$. The number of non-unit coefficients in an MS-Tree $\mathbf{M}$ is denoted as $\|\mathbf{M}\|$. We now offer a redundancy theorem about MS-Tree-based data representations.

THEOREM 2. *Any MS-Tree* $\mathbf{M}$*, in which at least one triplet contains more than one non-unit coefficients, is equivalent to (i.e., produces the same data approximation as) an MS-Tree* $\mathbf{M}'$*, such that every triplet* $C \in \mathbf{M}'$ *contains at most one non-unit coefficient, and* $\|\mathbf{M}'\| \leq \|\mathbf{M}\|$.

The proof, omitted due to space constrains, is similar in spirit to that of the analogous theorem for a Haar$^+$ tree [28]. The intuitive basis of this proof is a reduction of a triplet with two non-unit escort coefficients $q \neq 1$, $r \neq 1$ to one with only one non-unit main coefficient, with the appropriate change at its parent node, is shown in Figure 6.

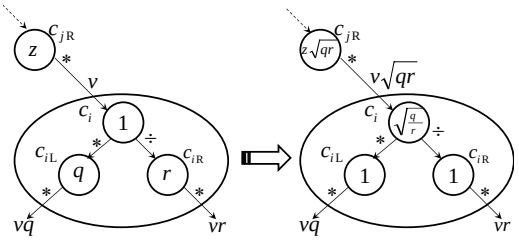


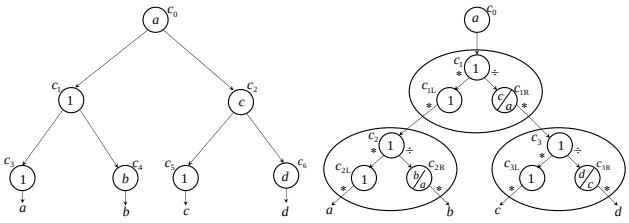
Figure 6: Triplet's Reduction Step

COROLLARY 1. *A* B*-term MS-Tree* $\mathbf{M}$ *that approximates a data vector* $\mathbf{D}$ *while minimizing an error metric* $\mathcal{E}$ *does not need to contain more than one non-unit coefficient value per triplet.*

In other words, it may be beneficial to use an escort coefficient instead of a main, but never more than one per triplet. An MS-Tree synopsis can stored non-unit coefficients succinctly, along with their indexes, grouped in three sets, one for each coefficient type; hence, the synopsis storage requires no extra bits to separately signify the type of each coefficient. In the next section, we present a dynamic-programming approximation scheme for MS-Tree summarization based on Corollary 1. Before we proceed, we provide the following theorem that proves the equivalence of a Compact Hierarchical Histogram, in its default *binary-tree* form [46], to an MS-Tree in which only escort coefficients (and the root) may assume non-unit values.

THEOREM 3. *A binary CHH (i.e., one in the default binary-tree form) with* B *non-zero nodes (bucket nodes) is equivalent to (i.e., achieves the same approximation as) an MS-Tree tree of* B *non-unit escort (or root) coefficients. In reverse, an MS-Tree with only* B *non-unit escort (or root) coefficients is equivalent to a* B*-term binary CHH.*

PROOF. Let $\mathbf{C}$ be a B-term binary CHH. For each non-zero term c_i in node i, let v_i be the value of the lowest occupied non-zero ancestor node. An MS-Tree $\mathbf{M}$ in which the escort coefficient corresponding to the position of each non-zero term c_i is assigned the value $\frac{c_i}{v_i}$ is equivalent to $\mathbf{C}$. The reverse equivalence follows with similar reasoning. □



(a) CHH (b) MS-Tree

Figure 7: CHH and equivalent MS-Tree

Figures 7 depicts an examples of the CHH/MS-Tree transformation used in Theorem 3. From Theorem 3 it follows that, as long as appropriate values can be assigned to MS-Tree nodes as coefficients, the MS-Tree can match the quality of a CHH. Moreover, an MS-Tree that uses main coefficients as well as the CHH-like escort ones should be able to achieve higher accuracy than a CHH under the same space budget.

6. MS-TREE SYNOPSIS CONSTRUCTION

PROBLEM 1. *Given a data vector* $\mathbf{D}$ *and a relative-error-based error function* $\mathcal{E}$*, construct an MS-Tree representation* $\mathbf{M}$ *of* $\mathbf{D}$ *with at most* B *non-unit coefficients that produces an approximation* $\hat{\mathbf{D}}$ *of minimal error* $f_{\mathcal{E}}\left(\left\|\mathbf{D} - \hat{\mathbf{D}}\right\|\right)$.

To solve this problem, we have to determine the optimal positions and values of the B non-unit coefficients. Since a triplet C_i needs to contain at most one non-unit coefficient, there are four available options in each of them. The following section provides a value delimitation framework for our solution.

6.1 Value Delimitation

Let v be an incoming value at the main node c_i of a triplet C_i, b be an amount of available synopsis space budget allocated to C_i and the subtree rooted in it and $E(i, v, b)$ the minimum achievable error, by a relative-error metric $\mathcal{L}_p^{\mathsf{rel}}$, in the subtree of C_i in that case. The overall minimum error can be derived in a bottom-up process that calculates $E(i, v, b)$ on each node, for each possible v and each amount of allocated space b. Let ℓ_i denote the layer in which C_i resides, counting from the bottom. Then the subtree of C_i contains 2^{ℓ_i} data leaves, hence needs no more than that synopsis space; thus, the domain of b for c_i is $D_i = \left\{0, 1, \ldots, \min\left\{B, 2^{\ell_i}\right\}\right\}$. The delimitation of possible incoming values v, as well as non-unit values assigned to the coefficients in C_i, is less straightforward. The examined values of v at C_i should match the possible outgoing values created by its parent triplet in the hierarchy. In order to maintain all such possible values within a finite domain, we need to work with the powers of a (small) base $1+\delta$, $\delta > 0$. Hence, the values we examine, $v \in \left\{(1+\delta)^k, k \in \mathrm{Z}\right\}$, become progressively more distant from each other. This disposition renders our multiplicative synopsis approach appropriate for relative-error summarization, as more emphasis is

paid on the accuracy of approximation for smaller absolute data values as opposed to larger ones. Moreover, instead of storing the actual real-valued coefficients, we need only store the *integer* logarithm k of a coefficient's value, rendering our synopses significantly more compact. We proceed to set upper and lower bounds on this domain of examined values, using the notation in Table 1.

Notation	*Denotation*
$\mathbf{D}$	Approximated data vector
$\mathbf{M}$	MS-Tree approximation of $\mathbf{D}$
C_i	Triplet of three coefficients (c_i, c_{iL}, c_{iR}) in $\mathbf{M}$
v	Incoming value to C_i
z_M	Value assigned to main coefficient of C_i
z_L (z_R)	Value assigned to c_{iL} (c_{iR})
z_0	Value assigned to root coefficient of $\mathbf{M}$
m_i (M_i)	Minimum (maximum) absolute value under C_i
m_L (m_R)	Minimum absolute value in left (right) subtree of C_i
M_L (M_R)	Maximum absolute value in left (right) subtree of C_i
m (M)	Global minimum (maximum) absolute value in $\mathbf{D}$
Λ	Ratio $\frac{M}{m}$
D_i	Domain of allocated space values b at C_i
$\mathcal{S}$	Set of potential incoming values v
$\mathcal{S}^v_{i,M}$	Set of pot. assigned values at the main coefficient of C_i
$\mathcal{S}^v_{i,L}$	Set of pot. assigned values at the left escort coeff. of C_i
$\mathcal{S}^v_{i,R}$	Set of pot. assigned values at the right escort coeff. of C_i

Table 1: Employed Notation

We start out with the following lemma, omitting its proof due to space constraints.

LEMMA 1. *For any incoming value $v > 0$ at a triplet C_i, there exist reconstructed values $\hat{d}_k$ and $\hat{d}_l$ in C_i's subtree such that $|\hat{d}_k| \leq v$ and $|\hat{d}_l| \geq v$.*

The following lemma defines the condition under which it may be useful to used a non-unit main coefficient.

LEMMA 2. *If the incoming value v at a triplet C_i is $v \notin (m_i, M_i)$, then the main coefficient c_i can be set to the unit value, i.e. $z_M = 1$, without deteriorating the quality of the approximation. Equivalently, $z_M \neq 1 \Rightarrow v \in (m_i, M_i)$.*

The proof, omitted due to space constraints, follows in lines similar to those used in the case of the Haar$^+$ tree in [28]. The following lemma derives from Lemma 2.

LEMMA 3. *An incoming value $v \in [m_i, M_i]$ at C_i results in at least as good approximation quality as an incoming value $v \notin [m_i, M_i]$, with the number of non-unit coefficients in the subtree of C_i being equal.*

The next theorem constrains the possible values assigned to a triplet's main coefficient c_i.

THEOREM 4. *Let $v \in (m_i, M_i)$ be the incoming value to C_i in $\mathbf{M}$. Then a non-unit value $z_M \neq 1$ assigned to c_i satisfies the inequality* $\min\left\{\frac{v}{M_R}, \frac{m_L}{v}\right\} \leq z_M \leq \max\left\{\frac{M_L}{v}, \frac{v}{m_R}\right\}$.

PROOF. C_i creates the outgoing pair $\left[vz_M, \frac{v}{z_M}\right]$. We first examine how small the (positive) magnitude of z_M can be. If $vz_M < m_L$ and $\frac{v}{z_M} > M_R$, then Lemma 3 implies that we can increase z_M so as to allow the outgoing value vz_M to reach m_L, or $\frac{v}{z_M}$ to reach M_R, or both, without deteriorating the quality of approximation. Hence, it should be:

$$vz_M \geq m_L \quad \vee \quad \frac{v}{z_M} \leq M_R \quad \Leftrightarrow \quad z_M \geq \min\left\{\frac{v}{M_R}, \frac{m_L}{v}\right\}$$

Likewise, if $vz_M > M_L$ and $\frac{v}{z_M} < m_R$, then, by Lemma 3, we can decrease z_M so as to allow vz_M to reach M_L, or $\frac{v}{z_M}$ to reach m_R, or both. Hence:

$$vz_M \leq M_L \quad \vee \quad \frac{v}{z_M} \geq m_R \quad \Leftrightarrow \quad z_M \leq \max\left\{\frac{M_L}{v}, \frac{v}{m_R}\right\}$$

In conclusion, $z_M \in \left[\min\left\{\frac{v}{M_R}, \frac{m_L}{v}\right\}, \max\left\{\frac{M_L}{v}, \frac{v}{m_R}\right\}\right]$. □

The application of Theorem 4 is problematic in case the minimum absolute value in a data set is zero. However, zero values can be treated separately in the synopsis construction, either by representing them by a bitmap and then omitting them, or by assigning part of the allocated space budget to them a priori and then solving the approximation problem for the rest of the data. Reasoning similar to that of Theorem 4 leads to the following theorem.

THEOREM 5. *A non-unit value $z_L \neq 1$ ($z_R \neq 1$) assigned to the left (right) escort coefficient at C_i satisfies the constraint $z_L \in \left[\frac{m_L}{v}, \frac{M_L}{v}\right]$ $\left(z_R \in \left[\frac{m_R}{v}, \frac{M_R}{v}\right]\right)$. Likewise, for a non-unit value $z_0 \neq 1$ assigned to the root coefficient, $z_0 \in [m, M]$.*

The next theorem constrains the incoming values to all triplets in $\mathbf{M}$ in terms of the global extrema m and M.

THEOREM 6. *The incoming value v to a triplet C_i in $\mathbf{M}$ satisfies the inequality $\frac{m}{\Lambda} < v < M\Lambda$, where $\Lambda = \frac{M}{m}$.*

The proof follows using Theorem 5, Lemma 2, and Theorem 4. Intuitively, Theorem 6 signifies that, in the worst case, a non-unit main coefficient c_i may need to cover the ratio $\frac{M}{m}$ of the two global extrema for one of the two outgoing values it creates and hence replicate this ratio in the other; for example, if c_i receives (almost) the extreme value M as incoming value, it may need to decrease it to m in one outgoing value, hence produce (almost) $\frac{M^2}{m}$ as its other outgoing value. In effect, the ratio of the largest to the lowest possible incoming value is Λ^3. Let $\mathcal{S} \subset \mathbb{R}$ denote the set of such values in $\left(\frac{m^2}{M}, \frac{M^2}{m}\right)$ that are powers of the working base $1+\delta$, $\mathcal{S} = \left\{(1+\delta)^k, k \in \mathbb{Z}\right\} \cap \left(\frac{m^2}{M}, \frac{M^2}{m}\right)$. Then $|\mathcal{S}| \leq 3\lfloor \log_{1+\delta} \Lambda \rfloor + 1 = O(\log_{1+\delta} \Lambda)$. According to Theorems 4 and 5, the same asymptotic bound $O(\log_{1+\delta} \Lambda)$ is valid for the cardinality of the sets $\mathcal{S}^v_{i,M} \subset \mathbb{R}$, $\mathcal{S}^v_{i,L} \subset \mathbb{R}$, $\mathcal{S}^v_{i,R} \subset \mathbb{R}$, containing the potential assigned values at the main, left, and right escort coefficient of a triplet C_i, respectively, that are powers of $1+\delta$, for a given incoming value v.

6.2 Computation of the Solution

We now develop a recursive dynamic-programming scheme for MS-Tree synopsis construction. Our approach is akin to those of [26, 8, 10, 22, 11, 15, 16, 36, 46, 28]. The algorithm operates in a bottom-up fashion over the MS-Tree, computing and tabulating all values of the $E(i, v, b)$ function. For each allowed incoming value v and each allocated space b to a triplet C_i, it examines all allowed distributions of b among the subtrees of C_i and all possible value assignments in C_i, and selects the best. To facilitate its operation, at each triplet C_i, it computes an array A from the pre-calculated arrays L and R of its children triplets C_{i_L} and C_{i_R}. The array entry $A[v, b]$ contains: (i) the best assigned value (power

of the base $1+\delta$) z_M, z_L, or z_R to assign at one of the coefficients in C_i, if any; (ii) the amount of space b_L out of b to allocate to the left subtree; and (iii) the minimum error $E(i,v,b)$ thus achieved. The size of A is $\left|\mathcal{S}\right| \cdot \left|D_i\right|$. $E(i,v,b)$ is computed as:

$$E(0,0,B) = \min_{z \in \mathcal{S}^0_{0,M}} \left\{ E\big(1, z, B-(z \neq 1)\big) \right\}$$

$$E(i,v,b) = \min \left\{ \begin{array}{l} \min\limits_{z_M \in \mathcal{S}^v_{i,M}, b' \in D_i} \left\{ \begin{array}{l} E\big(i_L, vz_M, b'\big) + \\ E\left(i_R, \frac{v}{z_M}, b-b'-(z_M \neq 1)\right) \end{array} \right. \\ \min\limits_{z_L \in \mathcal{S}^v_{i,L}, b' \in D_i} \left\{ \begin{array}{l} E\big(i_L, vz_L, b'\big) + \\ E\big(i_R, v, b-b'-(z_L \neq 1)\big) \end{array} \right. \\ \min\limits_{z_R \in \mathcal{S}^v_{i,R}, b' \in D_i} \left\{ \begin{array}{l} E\big(i_L, v, b'\big) + \\ E\big(i_R, vz_R, b-b'-(z_R \neq 1)\big) \end{array} \right. \end{array} \right. \quad (2)$$

The operations vz_X and $\frac{v}{z_M}$ are presented as such for illustration purposes. In fact, the algorithm does not need to conduct expensive multiplications and divisions, but can suffice itself to logarithmic operations, performing additions and subtractions in the realm of exponents k of the base $1+\delta$. For the sake of simplicity, error addition is used in Equation (2); in practice, any *distributive* error function, such as max, can be applied. For each of the three coefficients in a triplet C_i, Equation (2) computes the combination of a value assigned to that coefficient[2], if any, and a distribution of the available space b to the two subtrees rooted at C_i that achieves the least error in the subtree of C_i. It selects the least of these three computed minima as the value of $E(i,v,b)$. When a non-unit value is assigned to an examined coefficient, the remaining available space is decreased by one unit; this reduction is succinctly expressed by the boolean integer $(z_X \neq 1)$ in Equation (2). In the last tree layer, the optimal coefficient values are directly established by the data they approximate, by Theorem 1.

Complexity Analysis The size of the arrays A computed on each triplet C_i is $O\left(\log_{1+\delta} \Lambda \min\left\{B, \frac{n}{2^{\lfloor \log i \rfloor}}\right\}\right)$; besides, the scanning through all pairs of a potential assigned value in $\mathcal{S}^v_{i,M}$, $\mathcal{S}^v_{i,L}$, or $\mathcal{S}^v_{i,R}$ and an amount of allocated space in D_i takes $O\left(\log_{1+\delta} \Lambda \min\left\{B, \frac{n}{2^{\lfloor \log i \rfloor}}\right\}\right)$ time for each C_i and each $[v,b]$ pair. Thus, the running time is $O\left(\log^2_{1+\delta} \Lambda \sum_{i=1}^{n} \min\left\{B, \frac{n}{2^{\lfloor \log i \rfloor}}\right\}^2\right) = O\left(\log^2_{1+\delta} \Lambda n B\right)$. Besides, the algorithm can compute the minimum error for an $(1+\delta)$-base approximation without constructing the synopsis itself. As in [15], at most $\log n + 1$ arrays need be concurrently stored: one array for each triplet layer plus the arrays the algorithm is operating on at any instance. Hence the required space is $O\left(\log_{1+\delta} \Lambda \sum_{\ell=1}^{\log n} \min\left\{B, \frac{n}{2^\ell}\right\}\right)$ $= O\left(\log_{1+\delta} \Lambda B \log \frac{n}{B}\right)$, where ℓ is an MS-Tree triplet layer.

6.3 Options for Synopsis Construction

The construction of the actual synopsis after the minimum error result has been derived presents a time-space tradeoff. We may call the basic algorithm recursively for progressively smaller subproblems, starting with the two sub-trees of the first triplet C_1. Then the runtime is $O\left(\log^2_{1+\delta} \Lambda B \sum_{\ell=0}^{\log n} 2^\ell \frac{n}{2^\ell}\right)$ $= O\left(\log^2_{1+\delta} \Lambda n B \log n\right)$, hence a $\log n$ time factor is paid for the sake of space-efficiency.

On the other hand, with some necessary space-consuming book-keeping, the synopsis $\mathbf{M}$ can be extracted directly after the minimum error has been established. As in [28], this time-efficient approach, capable to operate in one pass over the data, has two variants: in case $B \gg \sqrt{n}$, it is advisable to keep all computed arrays in memory per se, requiring $O\left(\log_{1+\delta} \Lambda \sum_i \min\left\{B, \frac{n}{2^{\lfloor \log i \rfloor}}\right\}\right)$ $= O\left(\log_{1+\delta} \Lambda \sum_\ell 2^{\log n - \ell} \min\left\{B, 2^\ell\right\}\right) = O\left(\log_{1+\delta} \Lambda n \log B\right)$ space. Otherwise, in case if $B \ll \sqrt{n}$, it is advantageous to append to each array entry $A[v,b]$ the list of coefficient values forming the sub-solution represented by that entry, as in [16]. This option raises a $O\left(\log_{1+\delta} \Lambda \sum_\ell \min\left\{B, 2^\ell\right\}^2\right) = O\left(\log_{1+\delta} \Lambda B^2 \log \frac{n}{B}\right)$ space demand. The two options become balanced when $n \log B = B^2 \log\left(\frac{n}{B}\right) \Leftrightarrow B = \sqrt{n}$. Both cases are likely to appear in real-world applications.

6.4 The Case of Maximum Relative Error

In case the target error metric is the *maximum* relative error, we can employ a technique based on the dual problem, by analogy to [29]. Thus, we can utilize an algorithm that finds the minimum space required to satisfy a given maximum-relative-error bound, in a binary search iteration. This iteration converges to the $(1+\delta)$-optimal error value: an error value that cannot be possibly decreased without using more than B space. Given that maximum relative-error values equal to 1 can be trivially achieved by an all-zero synopsis, the value 1 can serve as the seed of the binary search. The binary search procedure will then converge after $O\left(\log \frac{1}{r}\right)$ iterations, where $0 < r \ll 1$ is the resolution with which the machine represents real numbers. We emphasize that this solution's dependence on r does *not* render it less exact. The algorithm of Section 6 that directly computes the exact $(1+\delta)$-optimal error result *also* computes it with a precision of r, since this is the precision with which the machine represents it anyway. Hence this *indirect* binary-search-based approach and the algorithm of Section 6 compute the $(1+\delta)$-optimal maximum relative error with the *same* precision on any given machine. The indirect approach needs only $O\left(\log^2_{1+\delta} \Lambda n \log \frac{1}{r}\right)$ time and $O\left(\log_{1+\delta} \Lambda \log n\right)$ space. The B factor has been eliminated, since the algorithm for the error-bounded problem does not feature a b parameter, as in [29]. Still, this algorithm requires multiple passes over the data, as the space-efficient direct algorithm does. In case that a solution in one-pass is required, than the two time-efficient direct options are preferable.

7. MULTIDIMENSIONAL EXTENSION

The efficient handling of multidimensional data is a major challenge for summarization algorithms. Thus, most past hierarchical summarization schemes [4, 10, 11, 16, 46] were extended to the multidimensional case. In this section, we propose a multidimensional extension of the MS-Tree and its related algorithms. Our task is to summarize a d-dimensional data array $\mathbf{D}$ with maximum domain size amongst its dimensions m, i.e., an array of $n = O(m^d)$ values.

7.1 Multidimensional Multiplicative Transform

The one-dimensional definition of the multiplicative transform (Section 4) can be extended in the multi-dimensional

[2] By Theorem 4, the main coefficient c_i is examined only if $v \in (m_i, M_i)$.

domain. The basic step of the decomposition in the two-dimensional case is shown in Figure 8(i). In this case, a two-dimensional array of four (positive) values $\{a, b, c, d\}$ is decomposed to the fourth root of their product $V = \sqrt[4]{abcd}$ and the *three* decomposition coefficients A, B, C holding the fourth root of different value ratios: $A = \sqrt[4]{\frac{ab}{cd}}$, $B = \sqrt[4]{\frac{ad}{bc}}$, $C = \sqrt[4]{\frac{ac}{bd}}$. The original data can be reconstructed using multiplication and division among the overall fourth root and the three decomposition coefficients: $a = VABC$, $b = \frac{VA}{BC}$, $c = \frac{VC}{AB}$, $d = \frac{VB}{AC}$. The figure depicts, along arrows, the set of three operations that have be performed between the *incoming value* V to a decomposition node and the three *stored coefficients* in that node, A, B, C respectively, in order to reconstruct the original data values. For the sake of comparison, the basic step of a (*non-standard*) two-dimensional Haar wavelet decomposition [4] is depicted in Figure 8(ii). In this case, the data are decomposed to their overall average value $V = \frac{a+b+c+d}{4}$ and three decomposition coefficients $A = \frac{a+b-c-d}{4}$, $B = \frac{a-b-c+d}{4}$, $C = \frac{a-b+c-d}{4}$; they can be reconstructed using addition and subtraction between the overall average and the three decomposition coefficients, as the figure shows. In the general, d-dimensional case, a tree node has at most 2^d children and the height of the tree is $\log_{2^d} m^d = \log m$ layers. Each non-root tree node contains at most $2^d - 1$ coefficients with the same support region, hence there are $O\left(\frac{n}{2^d}\right)$ nodes in the tree. The full multiplicative decomposition of a (two-dimensional) $m \times m$ array A is computed by recursively applying the basic decomposition step at successive levels of resolution. The overall fourth-root results V collected by quadruplets of values at one level play the role of data values in the next resolution level. The results of such a multiplicative transform can be arranged in an array by analogy to the (non-standard) DHWT [4]. The same process is extended to higher dimensionality. Still, our focus rests on developing a multidimensional definition of the MS-Tree, inspired by our definition of the multidimensional multiplicative transform.

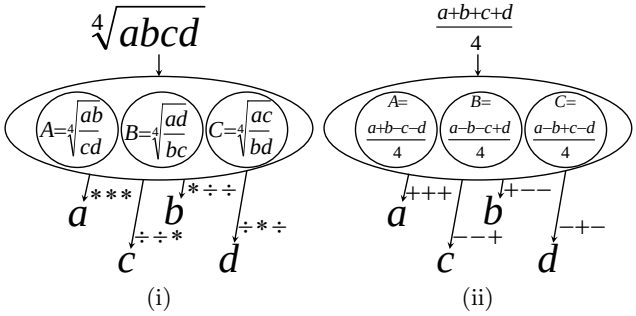


Figure 8: Two-dimensional Multiplicative Decomposition (i), and Haar wavelet decomposition (ii)

7.2 The Multidimensional MS-Tree

Figure 9 depicts a two-dimensional MS-Tree that can summarize a 16-element 4×4 two-dimensional data set. An MS-Tree node now has four (in general, 2^d for d dimensions) children nodes and contains *three* (in general, $2^d - 1$) main coefficients a, b, c, as well as *four* (in general, 2^d) escort coefficients q, r, s, t. These are combined by multiplication and division in order to create the *four* (in general, 2^d) outgoing values of that node, one towards each child node. Each child node summarizes a different region of the data array, called its *support region*. A node's main coefficients play the role of regular multiplicative transform coefficients and all share the same support region; each of the four escort coefficients is an additional factor on one of the four outgoing values, and shares the same support region as the main coefficients of its child node.

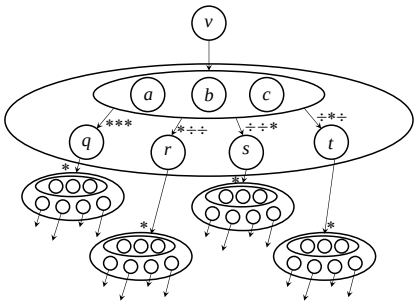


Figure 9: A two-dimensional MS-Tree

In the two-dimensional case, the *state* of a given node is an eight-element (in general, 2^{d+1}-element) vector $[v, a, b, c, q, r, s, t]$ containing the incoming value v to that node and the coefficient values a, b, c, q, r, s, t in it. A state of a node C *creates* the four-element (2^d-element) *outgoing vector* $\left[vqabc, \frac{vra}{bc}, \frac{vsc}{ab}, \frac{vtb}{ac}\right]$. Then the redundancy theorem about MS-Tree-based data representations can be extended to the two-dimensional case as follows.

THEOREM 7. *Any two-dimensional MS-Tree* $\mathbf{M}$*, in which at least one node contains more than three non-unit coefficients, is equivalent to an MS-Tree* $\mathbf{M}'$*, such that every node* $C \in \mathbf{M}'$ *contains at most three non-unit coefficients, and* $\|\mathbf{M}'\| \leq \|\mathbf{M}\|$.

PROOF. The basic idea is that an outgoing vector created by a node C_i with more than three non-unit coefficients can also be created by the same node in a state of at most three non-unit coefficients, and an adjustment of the parent node of C_i and proceeding recursively upwards. Details are omitted due to space constraints. □

Theorem 7 leads to the generalized form of Corollary 1:

COROLLARY 2. *A* B*-term* d*-dimensional MS-Tree* $\mathbf{M}$ *approximating data array* $\mathbf{D}$ *does not need to contain more than* $2^d - 1$ *non-unit coefficients per node.*

7.3 Multidimensional Value Delimitation

Lemma 1 is extended to the multidimensional case thanks to Theorem 7. The outgoing values of a node cannot be all greater (or less) than its incoming value v. Either v will equal at least one of the outgoing values, or it will be decreased in at least one outgoing value and increased in at least one other. Furthermore, we *postulate* the multidimensional version of Lemma 2: if the incoming value v at a node C_i is $v \notin (m_i, M_i)$, then all main coefficients in C_i are set to the unit value. In fact, we can also follow an approach that would render it provable, but that would require the addition of extra escort coefficients (one for each binary subdivision of a support region), incurring computational cost for marginal approximation benefits. We prefer to maintain simplicity. The postulate contains the value search space according to the following theorem.

THEOREM 8. *The incoming value v to a triplet C_i in* $\mathbf{M}$ *satisfies the inequality* $\frac{m^{2^d}}{M^{2^d-1}} < v < \frac{M^{2^d}}{m^{2^d-1}}$.

PROOF. Let v be an incoming value v to a node C_i outgoing from an ancestor triplet C_k of C_i, such that the incoming value v' to C_k itself is $v' \in (m, M)$. In the worst case, each of the $2^d - 1$ main coefficients in C_k will have the value $\frac{v'}{M}$ $\left(\frac{v'}{m}\right)$, in order to produce $2^d - 1$ outgoing values of the extreme value M (m), respectively; this will be the case as, for each of these $2^d - 1$ outgoing values, an even number of coefficients cancel each other out, hence v' is *divided* by the odd one out in each case. Assuming the worst case, let v be the *remaining* 2^dth outgoing value, i.e. the single one in which all coefficients contribute by multiplication, hence $v = \frac{v'^{2^d}}{M^{2^d-1}}$ $\left(\frac{v'^{2^d}}{m^{2^d-1}}\right)$. Still, $v' \in (m, M)$, hence $v \in \left(\frac{m^{2^d}}{M^{2^d-1}}, \frac{M^{2^d}}{m^{2^d-1}}\right)$. □

In effect, the ratio of the largest to the lowest possible incoming value is $\Lambda^{2^{d+1}-1}$, where $\Lambda = \frac{M}{m}$. Hence, in the d-dimensional case, $|\mathcal{S}| \leq \left(2^{d+1}-1\right)\lfloor \log_{1+\delta} \Lambda \rfloor + 1 = O(2^d \log_{1+\delta} \Lambda)$. On the other hand, the cardinality of the sets containing the potential assigned values at the main and escort coefficients of a triplet C_i that are powers of $1+\delta$ is $O(\log_{1+\delta} \Lambda)$.

7.4 General Multidimensional Algorithm

The one-dimensional algorithm of Section 6.2 can be extended to the multidimensional case. At each node C_i, it needs to consider $O\left(2^d \log_{1+\delta} \Lambda\right)$ incoming values; for each of those, it has to check $O\left(\left(\log_{1+\delta} \Lambda\right)^{2^d-1}\right)$ combinations of value assignments on $2^d - 1$ main coefficients, using the 2^d arrays returned from its children. For each tabulated value of available space b at node C_i with incoming value v, the algorithm needs to determine the *optimal distribution* of these b space units among the $2^d - 1$ main coefficients on C_i, its 2^d escort coefficients, and its 2^d children nodes. We can treat each escort coefficient as a member of the subtree at its child node. Hence, for each combination of values assigned to the main coefficients in C_i and amount of allocated space b at a child C_k of C_i rooted on escort coefficient c_e, we have to examine two cases: either b space is given to C_k and its subtree with incoming value v, or $b-1$ space is given to C_k, with a non-unit value z assigned to c_e and modifying v accordingly. The $(1+\delta)$-optimal value of z does *not* need to be separately computed for each v. Instead, it is computed only once for each b; thereafter, it is simply adjusted according to the given v, so as to produce the required best incoming value to C_k for the given value of b. The search for the optimal distribution of space b can be efficiently performed by ordering the children of C_i in a binary tree of $2^d - 1$ subnodes and executing binary search on them, as in [10, 11, 16]. This process takes $O\left(\log \min\left\{B, 2^{d\ell_i}\right\}\right)$ time per entry per subnode per combined value assignment, where ℓ_i is the MS-Tree layer of node C_i. Hence, the solution takes $O\left(2^{2d}\left(\log_{1+\delta} \Lambda\right)^{2^d} nB\right)$ time; only arrays of children nodes in a single root-to-bottom path need to be concurrently stored, hence, as there are at most 2^d children per node, the space is $O\left(\frac{2^{2d}}{d} \log_{1+\delta} \Lambda B \log \frac{n}{B}\right)$.

7.5 The Case of Maximum Relative Error

Still, in case that the target error function is the *maximum* relative error, we can do better. As in Section 6.4, we employ the algorithm that solves the *error-bounded* problem. We thus gain two advantages: First, we eschew the tabulation of space. Second, the cardinality of the set of possible incoming values is only $|\mathcal{S}| = O(\log_{1+\delta} \Lambda)$. This is due to the fact that, by Lemma 1, no examined incoming value needs to get too distant from the extrema of the data set; otherwise it would violate the *bound* on maximum relative error. The key operation is now a tabulation only for allowed incoming values at each node C_i. The algorithm determines the $(1+\delta)$-optimal assigned value for all main 2^d–1 coefficients residing on node C_i for each entry in this tabulation. We have to consider $O(\log_{1+\delta} \Lambda)$ incoming values at node C_i, and, for each of those, $O\left(\left(\log_{1+\delta} \Lambda\right)^{2^d-1}\right)$ combinations of value assignments on the $2^d - 1$ main coefficients in C_i, scanning through the 2^d arrays returned from children nodes. Since there are $O\left(\left(\frac{m}{2}\right)^d\right)$ nodes in the tree, the basic runtime becomes $O\left(2^d \left(\log_{1+\delta} \Lambda\right)^{2^d} \left(\frac{m}{2}\right)^d\right) = O\left(\left(\log_{1+\delta} \Lambda\right)^{2^d} n\right)$, and the space is $O\left(\frac{2^d}{d} \log_{1+\delta} \Lambda \log n\right)$. The tradeoff between time- and space-efficiency is treated as in the one-dimensional case. We utilize this algorithm in a binary-search iteration in order to solve the space-bounded problem. Thus we achieve $O\left(\left(\log_{1+\delta} \Lambda\right)^{2^d} n \log \frac{1}{r}\right)$ time and $O\left(\frac{2^d}{d} \log_{1+\delta} \Lambda \log n\right)$ space complexity. Remarkably, the benefit of this approach *increases* with dimensionality: a 2^{2d} time factor and a 2^d space factor are avoided. In Section 7.4, these factors derived from the cardinality of tabulated values (affecting time *and* space), and the need to compute a distribution of space units on children nodes (affecting time).

8. EXPERIMENTAL EVALUATION

This section presents our comparison of (one-dimensional) relative-error summarization with the following algorithms:

- **R-Haar** The restricted Haar wavelet synopsis algorithm of [11]. The Haar$^+$ model of [28], as well as the unrestricted Haar wavelet model of [16], can match the quality achieved with R-Haar, *subject to* a sufficiently small resolution of quantized values. Still, for practical resolution values, R-Haar may still produce competitive quality. Hence, we include this algorithm in our experimental study in order to assess this possibility. This algorithm provides an upper bound to the quality achieved with the probabilistic schemes of [10] and the streaming maximum-relative-error heuristics of [27].

- **CHH** The winning greedy CHH heuristic of [46]. The MS-Tree can match the quality of a CHH with sufficiently small resolution (Theorem 3), but CHH may still achieve competitive quality for practical resolution values. Thus, we include this algorithm in our study. Our implementation of the greedy heuristic in [46] takes into account the analysis of optimal-histogram bucket values for relative-error metrics in [22]. Our experimental study incidentally evaluates the performance of this CHH heuristic customized for relative

error in relation to other hierarchical synopsis techniques and to the optimal-relative-error-histogram algorithms of [22].

- **REHIST** The optimal-histogram construction algorithms for relative error metrics of [22]. The accuracy achieved with them constitutes an upper bound to the quality of approximate histogram techniques [44, 43, 5, 12, 17, 19, 13, 48].
- **Haar$^+$** The Haar$^+$ synopsis construction model of [28]. This model provides an upper bound to the quality achieved with its predecessor unrestricted Haar wavelet model of [16] under the same value quantization.
- **MSTree** Our MS-Tree algorithms of Section 6.

All algorithms were implemented with g++ 3.4.3, and experiments were run on a 2 CPU dual core Opteron 2.0GHz machine with 16GB of main memory running a 64Bit version of Fedora Core7. The following table summarizes the (basic) complexity requirements of these algorithms in the one-dimensional case, for the computationally more demanding $\mathcal{L}_2^{\text{rel}}$ metric; the fraction $\frac{\Delta}{\delta}$ expresses the cardinality of the set of examined values with the Haar$^+$ model.

Method	Time ($\mathcal{L}_2^{\text{rel}}$)	Space	Ref
R-Haar	$O(n^2 \log B)$	$O(n)$	[11, 15]
REHIST	$O(n^2 B)$	$O(n)$	[22, 15]
CHH	$O(nB^2 \log n)$	$O(B \log^2 n)$	[46]
Haar$^+$	$O\left(\left(\frac{\Delta}{\delta}\right)^2 nB\right)$	$O\left(\frac{\Delta}{\delta} B \log \frac{n}{B}\right)$	[28]
MS-Tree	$O(\log^2_{1+\delta} \Lambda nB)$	$O\left(\log_{1+\delta} \Lambda B \log \frac{n}{B}\right)$	this

Table 2: Complexity for one-dimensional synopsis construction

We have used two real-world data sets with hard to approximate bursts and discontinuities. The first data set (Aspect), also used in the studies of [10, 22, 28], is extracted from a relation of 581,012 tuples describing the forest cover type for 30 x 30 meter cells, obtained from US Forest Service. Aspect contains the frequencies of the distinct values of attribute `aspect` in the relation. The frequencies feature spikes of large values (min value: 499, max value: 6308). We have used a 256-value prefix of this data set. The second data set (Corel), also used in the performance study in [11], is a color histogram extracted from a Corel photo image collection. This histogram describes each Corel image in terms of 32 attributes corresponding to individual color densities (ranging from 10^{-6} to 1) for an 8×4 partitioning of the HSV color space. We made use of a 16384-value prefix of the first attribute from the color histogram feature (zero values were omitted). The data were downloaded from the UCI KDD Archive[3]. Our methods apply to much larger data sets too, but we confine ourselves to sizes with which all competitor methods can run in reasonable time.

8.1 Maximum Relative Error

In our first set of experiments, we present quality measurements for the maximum relative-error (MRE) metric, $\mathcal{L}_\infty^{\text{rel}}$. Figure 10 shows the results with the Aspect data set for increasing space budget B. A special sanity bound was not required with this data set, given that the minimum value in it is sufficiently large. In order to render the quality achieved with the MSTree directly and fairly comparable to that of the Haar$^+$ technique, we have defined the resolution of examined values with both of them in terms of the cardinality $\mathcal{S}$ of the arrays that contain such values. Hence both techniques require the same time and space resources. We show the results for $\mathcal{S} = 50$ and $\mathcal{S} = 100$. MSTree achieves consistently higher accuracy than all other models. Interestingly, Haar$^+$50 fails to preserve an advantage over REHIST, and is eventually outperformed by R-Haar as well. In fact, the increase of the space budget from $B = 40$ to $B = 60$ does not affect the quality with Haar$^+$50. This result is due to the coarse value resolution this algorithm works with; a similar behavior, in which the increasing space budget was not improving the quality of the synopsis measured in relative-error terms, was observed in the experimental study of [16] with the predecessor unrestricted Haar wavelet model. However, MSTree50 does not face such a problem, thanks to the logarithmic nature of that resolution and the multiplicative nature of the synopsis structure. Haar$^+$100 performs better, outperforming both REHIST and CHH, but fails to match its MSTree counterpart. CHH outperforms both R-Haar and REHIST with this dataset.

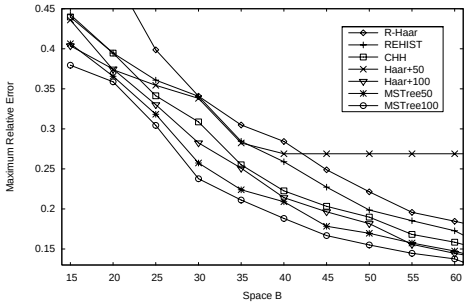


Figure 10: Quality comparison: Aspect, $\mathcal{L}_\infty^{\text{rel}}$

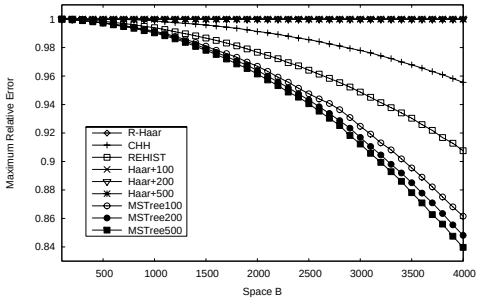


Figure 11: Quality comparison: Corel, $\mathcal{L}_\infty^{\text{rel}}$, $S = 10^{-6}$

Figure 11 shows the error results, with respect to the space budget B, with the Corel data set and the sanity bound set to the smallest value in the data set, $S = 10^{-6}$. Again, different value set cardinalities were tested with both MSTree and Haar$^+$ techniques, namely $\mathcal{S} = 100, 200, 500$. In this case, due to the hardness of approximation under the given sanity bound, both R-Haar and Haar$^+$ fail to achieve MRE values smaller than the worst-case value of 1, achieved by an all-zero synopsis. In contrast, the MSTree technique manages to extract synopses of non-trivial MRE values with all three tested cardinalities, and consistently outperforms both CHH and REHIST at that. A result of independent interest is that REHIST outperforms CHH with this data set; the flexibility of histogram buckets allowed for by REHIST en-

[3]Available at http://kdd.ics.uci.edu/

ables this method to focus on regions of the data vulnerable to high relative error in a way that the binary-hierarchy-based CHH model cannot. Although MSTree is also based on a fixed binary hierarchy, it has higher compression power thanks to its multiplicative nature.

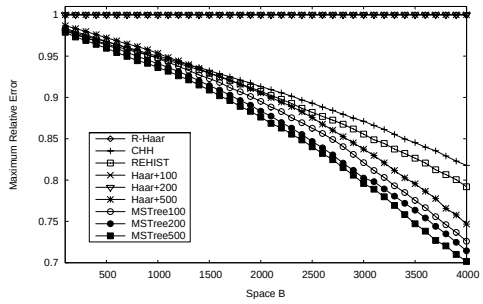


Figure 12: Quality comparison: Corel, $\mathcal{L}_\infty^{\text{rel}}$, $S = 0.01$

Figure 12 shows the error results for the same Corel data set with the sanity bound set at a higher value of $S = 0.01$. Now Haar$^+$500 manages to produce better-than-trivial synopses, outperforming both CHH and REHIST, thanks to the tolerance to errors for small values that the sanity bound implies. Still, Haar$^+$ with smaller cardinalities and R-Haar fail to produce better than worst-case accuracy. The picture with the other methods is similar to that of Figure 11, with all achieving smaller error values thanks to the higher sanity bound. MSTree is the best performer again.

8.2 RMS Relative Error

In our second set of experiments, we repeat our evaluation with the root-mean-squared relative-error (RMSRE) metric, $\mathcal{L}_2^{\text{rel}}$. Figure 13 presents the results with the Aspect data set. MSTree achieves the highest accuracy again. Likewise, Haar$^+$50 is matched by and eventually outperformed by R-Haar, while Haar$^+$100 fares better, outperforming both REHIST and CHH, but still does not match its MSTree counterpart. Interestingly, CHH can again outperform REHIST with this dataset, and stands between the two Haar$^+$ variants.

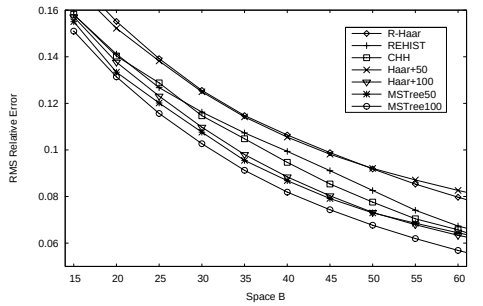


Figure 13: Quality comparison: Aspect, $\mathcal{L}_2^{\text{rel}}$

Figure 14 depicts the results with the Corel data set and sanity bound $S = 10^{-6}$; results for $\mathcal{S} = 500$ are not presented for the sake of readability. Despite the aggregate nature of the RMSRE metric, as opposed to MRE, R-Haar produces worst-case accuracy. The two Haar$^+$ variants fare better, but are still outperformed by all other methods except R-Haar. REHIST outperforms CHH, but MSTree is the best performer.

Figure 15 shows the results with $S = 0.01$. R-Haar just overcomes worst-case performance. Haar$^+$ is second worst, while the gap between REHIST and CHH is narrowed. MSTree achieves the highest accuracy again.

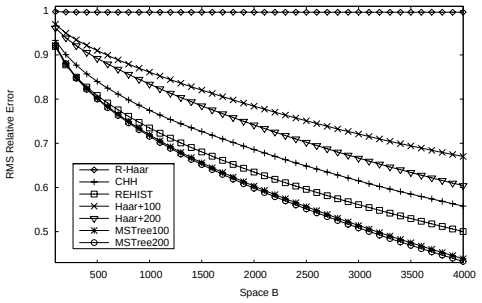


Figure 14: Quality comparison: Corel, $\mathcal{L}_2^{\text{rel}}$, $S = 10^{-6}$

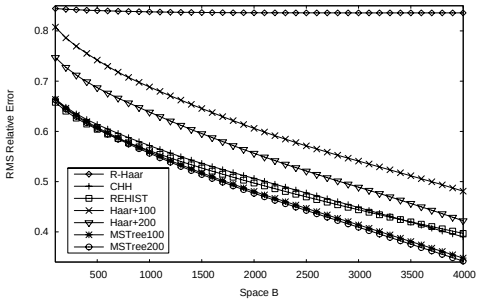


Figure 15: Quality comparison: Corel, $\mathcal{L}_2^{\text{rel}}$, $S = 0.01$

8.3 Time Comparison

Table 2 indicates that MS-Tree performs at least as well or better than its competitor hierarchical techniques in terms of time for synopsis construction; besides, they are all based on the same type of structure. Thus, it is mostly interesting to compare the MS-Tree to the structurally and algorithmically different REHIST method in terms of *time* for synopsis construction. We have done so with the more computationally demanding RMS relative-error minimization algorithms on different-sized collections of attribute values from the Corel data set, while setting the value set cardinality for MSTree at $\mathcal{S} = 50$ and $\mathcal{S} = 100$, and the space budget at $B = 100$. Figure 16 shows the results on logarithmic axes. Expectedly, the time for MS-Tree synopsis construction grows linearly in n, while REHIST has quadratic growth.

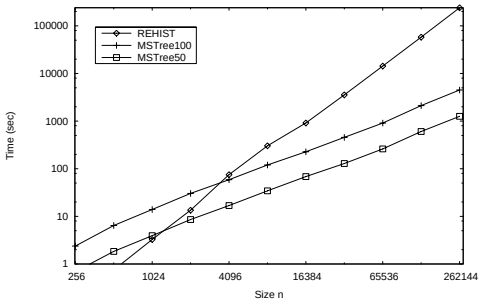


Figure 16: Time comparison: Corel, $\mathcal{L}_2^{\text{rel}}$, $S = 10^{-6}$

9. CONCLUSIONS

In this paper we have introduced multiplicative summarization: a novel data reduction technique that addresses the needs for effective relative-error oriented summarization. This method inherits from past hierarchical synopsis techniques but alters their basic mechanism, using multiplication as the basic data reconstruction tool. We have shown that this approach allows for proportional attention to be

paid to those parts of the data which are more susceptible to high relative errors, that is data of small absolute value. We proposed the MS-Tree as a multiplicative summarization structure and developed efficient dynamic programming algorithms for synopsis construction based on it. We generalized our results to any data dimensionality, and we have shown how to efficiently minimize the *maximum* relative error in the multidimensional case. We have conducted the *first*, to our knowledge, experimental comparison of state-of-the-art hierarchical and optimal-histogram summarization techniques for relative-error-based metrics. Therewith, we demonstrated that the MS-Tree outperforms previous models at hard relative-error summarization problems. Besides, MS-Tree synopsis construction runs in time linear in the size of the data and can be performed in one pass. In conclusion, our solutions provide a mostly recommendable option for the high quality and time-efficient relative-error summarization of very large discontinuous data sets. In the future, we intend to examine how the multiplicative paradigm can be applied to other models, such as the GenHist model.

10. REFERENCES

[1] A. Aboulnaga and S. Chaudhuri. Self-tuning histograms: building histograms without looking at data. In *SIGMOD*, 1999.

[2] D. Agarwal, D. Barman, D. Gunopulos, N. E. Young, F. Korn, and D. Srivastava. Efficient and effective explanation of change in hierarchical summaries. In *KDD*, 2007.

[3] N. Bruno, S. Chaudhuri, and L. Gravano. STHoles: a multidimensional workload-aware histogram. In *SIGMOD*, 2001.

[4] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. *VLDB Journal*, 10(2-3):199–223, 2001 (also VLDB 2000).

[5] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *TODS*, 27(2):188–228, 2002 (also SIGMOD 2001).

[6] S. Chen and A. Nucci. Dynamic nonuniform data approximation in databases with Haar wavelet. *Journal of Computers*, 2(8):64–76, 2007 (also DCC 2007).

[7] G. Cormode, M. Garofalakis, and D. Sacharidis. Fast approximate wavelet tracking on streams. In *EDBT*, 2006.

[8] A. Deligiannakis, M. Garofalakis, and N. Roussopoulos. Extended wavelets for multiple measures. *TODS*, 32(1), 2007 (also SIGMOD 2003).

[9] A. Deshpande, M. Garofalakis, and R. Rastogi. Independence is good: Dependency-based histogram synopses for high-dimensional data. In *SIGMOD*, 2001.

[10] M. Garofalakis and P. B. Gibbons. Probabilistic wavelet synopses. *TODS*, 29(1):43–90, 2004 (also SIGMOD 2002).

[11] M. Garofalakis and A. Kumar. Wavelet synopses for general error metrics. *TODS*, 30(4):888–928, 2005 (also PODS 2004).

[12] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. *TODS*, 27(3):261–298, 2002 (also VLDB 1997).

[13] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *STOC*, 2002.

[14] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Optimal and approximate computation of summary statistics for range aggregates. In *PODS*, 2001.

[15] S. Guha. On the space-time of optimal, approximate and streaming algorithms for synopsis construction problems. *VLDB Journal*, accepted for publication, 2009 (also VLDB 2005).

[16] S. Guha and B. Harb. Approximation algorithms for wavelet transform coding of data streams. *IEEE Transactions on Information Theory*, 54(2):811–830, 2008 (also SODA 2006).

[17] S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Histogramming data streams with fast per-item processing. In *ICALP*, 2002.

[18] S. Guha, C. Kim, and K. Shim. XWAVE: Approximate extended wavelets for streaming data. In *VLDB*, 2004.

[19] S. Guha, N. Koudas, and K. Shim. Approximation and streaming algorithms for histogram construction problems. *TODS*, 31(1):396–438, 2006.

[20] S. Guha, N. Koudas, and D. Srivastava. Fast algorithms for hierarchical range histogram construction. In *PODS*, 2002.

[21] S. Guha, H. Park, and K. Shim. Wavelet synopsis for hierarchical range queries with workloads. *VLDB Journal*, 17(5):1079–1099, 2008.

[22] S. Guha, K. Shim, and J. Woo. REHIST: Relative error histogram construction algorithms. In *VLDB*, 2004.

[23] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Selectivity estimators for multidimensional range queries over real attributes. *The VLDB Journal*, 14(2):137–154, 2005 (also SIGMOD 2000).

[24] Y. E. Ioannidis. Universality of serial histograms. In *VLDB*, 1993.

[25] H. V. Jagadish, H. Jin, B. C. Ooi, and K.-L. Tan. Global optimization of histograms. In *SIGMOD*, 2001.

[26] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *VLDB*, 1998.

[27] P. Karras and N. Mamoulis. One-pass wavelet synopses for maximum-error metrics. In *VLDB*, 2005.

[28] P. Karras and N. Mamoulis. The Haar$^+$ tree: a refined synopsis data structure. In *ICDE*, 2007.

[29] P. Karras, D. Sacharidis, and N. Mamoulis. Exploiting duality in summarization with deterministic guarantees. In *KDD*, 2007.

[30] S. Khanna, S. Muthukrishnan, and S. Skiena. Efficient array partitioning. In *ICALP*, 1997.

[31] N. Koudas, S. Muthukrishnan, and D. Srivastava. Optimal histograms for hierarchical range queries. In *PODS*, 2000.

[32] J.-H. Lee, D.-H. Kim, and C.-W. Chung. Multi-dimensional selectivity estimation using compressed histogram information. In *SIGMOD*, 1999.

[33] Y. Matias and D. Urieli. Optimal workload-based weighted wavelet synopses. *Theoretical Computer Science*, 371(3):227–246, 2007 (also ICDT 2005).

[34] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. In *SIGMOD*, 1998.

[35] M. Muralikrishna and D. J. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *SIGMOD*, 1988.

[36] S. Muthukrishnan. Subquadratic algorithms for workload-aware Haar wavelet synopses. In *FSTTCS*, 2005.

[37] S. Muthukrishnan, V. Poosala, and T. Suel. On rectangular partitionings in two dimensions: Algorithms, complexity, and applications. In *ICDT*, 1999.

[38] S. Muthukrishnan and M. Strauss. Rangesum histograms. In *SODA*, 2003.

[39] S. Muthukrishnan, M. Strauss, and X. Zheng. Workload-optimal histograms on streams. In *ESA*, 2005.

[40] S. Muthukrishnan and T. Suel. Approximation algorithms for array partitioning problems. *Journal of Algorithms*, 54(1):85–104, 2005.

[41] G. Piatetsky-Shapiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. In *SIGMOD*, 1984.

[42] V. Poosala, V. Ganti, and Y. E. Ioannidis. Approximate query answering using histograms. *IEEE Data Eng. Bull.*, 22(4):5–14, 1999.

[43] V. Poosala and Y. E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *VLDB*, 1997.

[44] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *SIGMOD*, 1996.

[45] L. Qiao, D. Agrawal, and A. E. Abbadi. RHist: adaptive summarization over continuous data streams. In *CIKM*, 2002.

[46] F. Reiss, M. Garofalakis, and J. M. Hellerstein. Compact histograms for hierarchical identifiers. In *VLDB*, 2006.

[47] U. Srivastava, P. J. Haas, V. Markl, M. Kutsch, and T. M. Tran. ISOMER: Consistent histogram construction using query feedback. In *ICDE*, 2006.

[48] E. Terzi and P. Tsaparas. Efficient algorithms for sequence segmentation. In *SIAM SDM*, 2006.

[49] N. Thaper, S. Guha, P. Indyk, and N. Koudas. Dynamic multidimensional histograms. In *SIGMOD*, 2002.

[50] J. S. Vitter and M. Wang. Approximate computation of multidimensional aggregates of sparse data using wavelets. In *SIGMOD*, 1999.