Approximate Query Processing in Cube Streams

Ming-Jyh Hsieh, Ming-Syan Chen, *Fellow*, *IEEE*, and Philip S. Yu, *Fellow*, *IEEE*

Abstract—Data cubes have become important components in most data warehouse systems and decision support systems. In such systems, users usually pose very complex queries to the Online Analytical Processing (OLAP) system, and systems usually have to deal with a huge amount of data because of the large dimensionality of the sets; thus, approximating query processing has emerged as a viable solution. Specifically, the applications of cube streams handle multidimensional data sets in a continuous manner in contrast to the traditional cube approximation. Such an application collects data events for cube streams online, generates snapshots with limited resources, and keeps the approximated information in a synopsis memory for further analysis. Compared to the OLAP applications, applications of cube streams are subject to many more resource constraints on both the processing time and the memory and cannot be dealt with by existing methods due to the limited resources. In this paper, we propose the DAWA algorithm, which is a hybrid algorithm of Discrete Cosine Transform (DCT) for Data and the discrete wavelet transform (DWT), to approximate cube streams. Our algorithm combines the advantages of the high compression rate of DWT and the low memory cost of DCT. Consequently, DAWA requires much smaller working buffer and outperforms both DWT-based and DCT-based methods in execution efficiency. Also, it is shown that DAWA provides a good solution for an approximate query processing of cube streams with a small working buffer and a short execution time. The optimality of the DAWA algorithm is theoretically proved and empirically demonstrated by our experiments.

Index Terms—Cube streams, OLAP, data cubes, data streams.

1 Introduction

A large number of data warehouses and data cubes have been constructed and deployed in applications since the concepts of Online Analytical Processing (OLAP) techniques and data cubes were introduced in [1]. In addition, data cubes have become important components in most data warehouse systems and decision support systems. Answering range queries is one of the primary tasks of the OLAP applications. For example, when exploring marketing databases, users may be interested in discovering the total purchases of several products at all branches of a company in California during the last three days. In such systems, users usually pose very complex queries to the OLAP system, which requires complex operations over gigabytes of data and takes a very long time to produce exact answers. Therefore, approximate query processing has recently emerged as a viable solution for dealing with the huge amount of data.

In addition, some applications such as phone call analysis by cellular phone companies or sales volume monitoring in retailers need a faster mechanism for processing continuously incoming information. Such applications handle stream data in a multidimensional form rather than the static form used in traditional OLAP applications. The basics of data streams on approximating frequency counts [2], temporal aggregations [3], maintaining statistics [4], and further analyzing techniques [5], [6] are explored in previous works. However, for those multidimensional data streams or cube streams, OLAP queries require complex operations over gigabytes of data and take a very long time to produce exact answers. Thus, an approximate query processing over cube streams is a viable solution. Also, the volume of data is usually too huge to be stored in permanent devices or be scanned thoroughly more than once. It is recognized, therefore, that both approximation and adaptability are the key requirements for executing queries over rapid multidimensional data streams. Such applications collect data events for multidimensional data sets (MDSs) online, generate snapshots with limited resources, and keep the approximated information in a synopsis memory for further analysis. In other words, the applications generate snapshots of cube streams (SCSs). Like the OLAP applications, the SCS ones have a number of options such as a user may request a snapshot in a few seconds rather than wait for the exact answer, which could take tens of minutes to compute [7]. However, the SCS applications differ from the traditional OLAP applications in two important aspects. First, the resources for both the processing time and the memory are much more constrained than in offline cube construction; that is, cube streams must be processed efficiently with a small working buffer to deal with the rapid growth of data events. Second, the data events are time relevant in nature; thus, it is appropriate to model them as brown noise [8]. As a result, an efficient algorithm that can compress multidimensional data streams within a small working buffer in one data scan is required to address such a problem.

In this paper, we explore both the generation of snapshots for *cube streams* in a small working buffer with one

- *M.-J. Hsieh and M.-S. Chen are with the Department of Electrical Engineering, National Taiwan University, 106, No. 1, Sec. 4 Roosevelt Road, Taipei, Taiwan, ROC. E-mail: mintz@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw.*
- *P.S. Yu is with the IBM T.J. Watson Research Center, PO Box 704, Yorktown, NY 10598. E-mail: psyu@us.ibm.com.*

Manuscript received 26 Feb. 2006; revised 29 July 2006; accepted 22 Jan. 2007; published online 7 June 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0100-0206.
Digital Object Identifier no. 10.1109/TKDE.2007.190622.

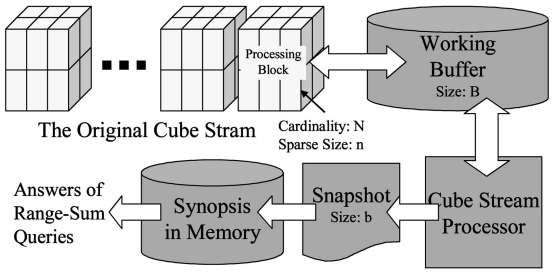


Fig. 1. The system model for processing multidimensional data streams.

data scan and also the storage of the snapshots in synopsis memory. Since the original data sets grow continuously, snapshots should be captured periodically for each given time slot and processed rapidly. Fig. 1 shows an extension of the computational model proposed in [9].

In SCS applications, the data events are collected progressively and are processed in the working buffer until the predefined time limit is reached. The subblock segmented in the time interval of a cube stream is called a processing block (PB). Note that the synopsis memory is designed to store the snapshots over a long period; therefore, the processed data in the working buffer should be compressed further to keep as many snapshots of PBs as possible. Hence, the size of the working buffer, denoted as B, is usually much smaller than the size of a PB, denoted as N, so the size of the snapshot of each PB, denoted as b, should be much smaller than B. For each PB, the data set can be regarded as a set of static OLAP data cubes, so the techniques for maintaining the aggregations of OLAP cubes can be applied. It is noted that the traditional OLAP approaches suffer from several critical problems such as requiring long processing time and large working buffer to process a PB and cannot comply with the requirements of the SCS applications.

1.1 Related Works

To deal with the applications of cube streams, we devise an algorithm to approximate PBs. For static OLAP cubes, several approaches have been developed to compress data efficiently in order to solve the problems of selectivity estimation [10], [11] and approximating query processing [7]. Most of these works are based on either the Discrete Wavelet Transform (DWT) or the Discrete Cosine Transform (DCT). The DWT-based approaches use the Haar wavelets, which is a mathematical tool for hierarchical decomposition of functions with several successful applications in signal and image processing [12], [13]. Meanwhile, the DCT-based approaches have been widely used in image and signal processing in 2D domains, since they can take advantage of the unequal frequency distribution of natural signals and provide acceptable compression efficiency. It has been shown empirically that these two approaches can compress original data sets into a small number of coefficients and provide answers of acceptable quality for range-sum queries. However, several limitations, which we will describe later, prevent their use in SCS applications.

It has been shown that Haar wavelets can provide answers of acceptable quality for range-sum queries with a high compression rate, as long as the working buffer is large enough. The problem of applying wavelets to SCS applications is that the corresponding memory cost, which is as high as the cardinality of a PB, limits the volume of data to less than or equal to the size of the working buffer. Consequently, one can only segment the original cube stream into small PBs for processing in the limited working buffer, at the cost of degrading the quality of the answers. An alternative approach adopts a thresholding technique to accommodate larger PBs [11]. By dropping small coefficients between each round of decomposition, this technique reduces the memory required to the size of a working buffer. However, the quality of answers is also degraded significantly. Vitter and Wang [7] take the advantage of the sparseness of real-world data sets and store temporal results in a secondary storage to avoid the shortcomings of thresholding. However, the intensive I/O between the working buffer and the secondary storage makes this approach impractical for SCS applications. Moreover, all the DWT-based approaches suffer from the problems of dimension ordering and the unnecessary decomposition of null cells (zeros). The disadvantages of DWT, including the problems of small PBs and coefficient thresholding, will be discussed further in Section 2.

The work in [10] propose a DCT-based approach that compresses information from a large number of small-sized histogram buckets by using the DCT. A mechanism for selecting the best set of theoretical DCT coefficients for answering range-sum queries is also proposed. The transformation of each DCT coefficient is independent, so this approach is able to work in a small amount of main memory. However, its time complexity, estimated as $O(nB)$, is too high for SCS applications to process PBs online.

1.2 Our Contributions

In this paper, we propose the DAWA algorithm, which is an integrated algorithm of DCT for Data and DWT, for approximating the cube streams. The algorithm compresses PBs in two phases. The first phase, called the DCT phase, partitions a PB into several subblocks and then applies DCT on each of them by reciprocal zonal sampling. The second phase, called the DWT phase, retrieves the DCT coefficients located at the same position of all the subblocks and joins them into several series. The series shown to be optimal are decomposed by the Haar wavelets to further reduce the space required in order to save the information to the synopsis memory. Also, several heuristics are applied to improve the accuracy of reconstruction. The superiority of the DAWA algorithm over other approaches lies in its ability to integrate the high compression rate of DWT and the low memory cost of DCT, thus enabling DAWA to provide answers of good quality for SCS applications with a small working buffer and execution time. The optimality of the algorithm is first proved theoretically and empirically demonstrated by our experiments.

The contributions of this paper are manifold. We address the problem of approximating cube streams. Today's DSS applications require faster techniques to handle continuously growing data cubes. The problem of handling those multidimensional data streams is formulated in this paper. We propose an efficient algorithm and several heuristics to approximate cube streams with very restricted resources. In addition, the proposed algorithm improves traditional

OLAP applications to obtain faster and more accurate results.

The remainder of this paper is organized as follows: Preliminaries are presented in Section 2. The theoretical basis and detailed steps of the DAWA algorithm are introduced in Section 3. Empirical studies are conducted in Section 4. Finally, in Section 5, we present our conclusions.

2 Preliminaries

In this section, we discuss several techniques for solving the problems of approximating cube streams under a limited working buffer and synopsis memory. An MDS consists of categorical attributes, for example, Site_type and Site_Region, which may serve as the dimensions, and numeric attributes, for example, number of errors, which serve as the measures. Also, an MDS constructed with multidimensional data streams is usually partitioned into blocks with a predefined time period T. It is then processed individually for snapshot generation, since the working buffer is constrained. Partitioned blocks, regarded as OLAP cubes, are referred to as *PBs*.

In general, the above measures are aggregated to the combination of dimension attributes using functions like sum, average, count, and variance. An important class of aggregation query is the range-sum query, which applies SUM operations over a set of continuous data cells [7]. For range-sum queries of the cube stream, the answers can be estimated by summing up the measures of all cells at each PB. Without loss of generality, we focus on the problem of estimating the subtotal of a PB. The Haar wavelets and its extension to MDSs are described in Section 2.1. The basics of DCT, including the reciprocal zonal sampling and integral techniques for answering range-sum queries, are introduced in Section 2.2.

2.1 The Haar Wavelets

The wavelet represents a function in terms of a coarse overall approximation and a series of detailed coefficients that revise the function from coarse to fine. Wavelets are constructed in various forms such as orthonormal [14] and nonorthogonal [15]. In the modern wavelet theory, the Haar filter is the foundation of wavelets. This technique is advantageous because of its linear computational complexity of $O(N)$, which is ideal for data streams.

2.1.1 Wavelet Decomposition and Thresholding

The concept of wavelet transform can be best understood by the following example:

Example 1. Suppose there are eight values collected at some moment that form a numerical time series $S = \{64, 48, 16, 32, 56, 56, 48, 24\}$. For a multiresolution analysis, the values are pairwisely averaged to get a low-resolution signal first. Therefore, we have {56, 24, 56, 36}, where the first two values in the original signal, that is, 64 and 48, are averaged to 56, the second two values 16 and 32 are averaged to 24, and so on. To avoid losing any information in this averaging process, the difference values, which are

TABLE 1
Wavelet-Based Multiresolution Analysis

Resolution	Averages	Differences
8	{64, 48, 16, 32, 56, 56, 48, 24}	-
4	{56, 24, 56, 36}	{8, -8, 0, 12}
2	{40, 46}	{16, 10, 8, -8, 0, 12}
1	{43}	{-3, 16, 10, 8, -8, 0, 12}

$$\{8(=64-56), -8(=16-24), 0(=56-56), 12(=48-36)\},$$

should also be stored. As such, the original values can be reconstructed from these average and difference values.

Consequently, the original signal is successfully decomposed into a low-resolution version of half the number of values with a corresponding set of difference values. By performing this process recursively, the full decomposition can be obtained, as shown in Table 1.

Note that wavelet coefficients, which correspond to different resolution scales, are generated recursively. The decomposed wavelet coefficients for original series are

$$\widehat{S} = \{43, -3, 16, 10, 8, -8, 0, 12\}.$$

The Haar wavelets decomposition can be extended to multidimensional data arrays by performing a series of one-dimensional decomposition. For example, in the 2D case, we first apply one-dimensional decomposition to each row of data. Next, the transformed rows are treated as the original data set, and the decomposition to each column is performed. The detailed steps of multidimensional decompositions can be found in [7], [11].

Also, the Haar wavelet has been proven to be efficient for dealing with MDSs, as long as the working buffer is large enough to accommodate the entire data cube. Clearly, a trade-off between quality and space cost has to be considered. For SCS applications, the available space for the working buffer is usually much smaller than the size of the PB, that is, $B \ll N$. Thus, only some of the coefficients, being representative in general, can be kept, and the rest must be pruned. This technique, called thresholding, selects the top-k coefficients with the largest absolute values and yields the minimum L_2-norm error [16], [17]. This approach is widely adopted for thresholding.

2.1.2 Approximating Processing Blocks

Thresholding provides a high compression rate and quality answers if it is applied after the decomposition along all dimensions is finished. However, the requirement for the working buffer is infeasible. Instead, thresholding is performed between two successive decompositions. This, however, compromises the optimality of global thresholding. As a result, the quality of the reconstruction is affected significantly. Moreover, the results of decomposition depend on the dimension order. Hence, the best selection of the dimension order needs to be evaluated empirically. This problem can be best understood by the following example:

Example 2. Consider the 2D data set in Fig. 2a. The size of each dimension is 8. By applying multidimensional wavelet decomposition techniques [7] to this data set, decompositions along both dimensions are performed iteratively. The reconstruction results are shown in Fig. 2,

(a)

	0	1	2	3	4	5	6	7
0	81	89	59	95	77	30	105	43
1	123	116	57	94	17	57	113	36
2	121	109	114	98	86	9	79	116
3	83	**115**	**19**	**19**	**72**	**21**	**68**	24
4	26	73	33	122	42	28	92	34
5	51	36	17	28	28	34	14	77
6	73	50	104	44	33	45	91	107
7	93	30	18	79	109	105	124	23

55	55	75	75	56	75	65	65
55	55	75	75	56	75	65	65
75	75	95	95	114	56	85	85
76	76	16	16	36	55	46	46
65	65	24	106	73	73	37	78
65	65	69	62	73	73	37	78
65	65	84	47	43	43	83	93
65	65	84	47	73	73	104	12

(b)

58	88	58	58	65	65	79	51
58	88	58	58	65	65	79	51
58	88	82	34	65	65	56	74
58	88	34	82	65	65	102	28
56	48	55	102	65	65	99	32
56	48	102	55	65	65	60	71
89	37	67	67	31	31	79	51
89	37	67	67	99	99	79	51

(c)

Fig. 2. An example of an 8×8 data set and the reconstruction results. (a) An example of an $S \times S$ data set. (b) Along x-axis first. (c) Along y-axis first.

Fig. 2b is the result decomposed along the x-axis first from Fig. 2a, and Fig. 2c is that decomposed along the y-axis first. The percentage of coefficients to be retained in each step is set at 25 percent.

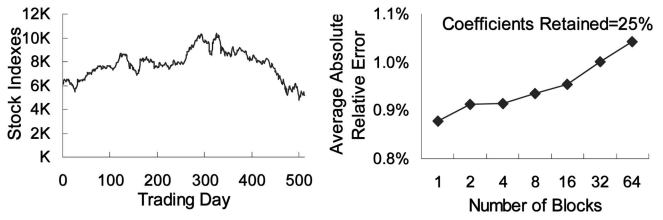
If a range-sum query $Q_s = Sum(3:3, 1:6)$ requesting the summation of the cells with bold numbers is submitted, the answer calculated in Fig. 2b will be very different from that in Fig. 2c. The former gives 245, and the latter gives 436, whereas the correct answer is $115 + 19 + 19 + 72 + 21 + 68 = 314$, showing that wavelet decomposition is degraded by the order dependency imposed on the corresponding reconstruction results.

Example 2 shows that the results of decomposition depend on the dimension order. The best dimension order cannot be decided, unless all possible sequences are evaluated. However, there are $N!$ possible sequences for an N-dimensional data cube. Clearly, evaluating all results for possible dimension order is not practical for real-world applications. In addition, the computational complexity is proportional to the cardinality N of the PB. For real-world data sets, the number of nonzero data points, denoted as n, is much smaller than N [18], [19]. Therefore, it is inefficient to perform the Haar wavelets decomposition on sparse data sets.

2.1.3 The Effect of Small Blocks

To avoid interlaced decomposition and thresholding, the size of a PB block must be reduced to fit the working buffer. However, it takes much more time to compute the answers for range-sum queries from many small blocks. Moreover, the selection of snapshot coefficients may not be optimal due to the uneven distribution of different partitions; that is, the size for a snapshot may be too large or too small for different PBs. Example 3 explains the problem caused by small PBs.

Example 3. Fig. 3a shows the data sequence for a real data set consisting of stock indices during 512 successive trading days. If only 25 percent of the coefficients can be retained for each block, the multiresolution decomposition will cause the 0.8 percent error.



(a) (b)

Fig. 3. (a) The indices chart. (b) The average absolute relative error for various segments, where 25 percent of the coefficients (128) are retained.

It is shown in Fig. 3b that the average absolute relative error[1] increases as the number of split segments increases.

Due to the shortcomings of thresholding and small blocks, DWT is not appropriate for approximating cube streams directly. To remedy this, we describe a new approach to compress a PB into a more compact form in Section 3.

2.2 The DCT for Data

The DCT has been widely used in the image and signal processing because of its capability of compressing information. Recent works have extended the technique to compress multidimensional histogram [10] and OLAP cubes. For a series of data $\overrightarrow{f} = (f(0), f(1), \ldots f(N-1))$, DCT coefficients $\overrightarrow{F} = (F(0), F(1), \ldots F(N-1))$ are defined as

$$F(u) = \sqrt{\frac{2}{N}} c_u \sum_{n=0}^{N-1} f(n) \cos\left(\frac{(2n+1)u\pi}{2N}\right),$$

where u is the frequency index, and

$$c_u = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u = 0 \\ 1 & \text{for } u \neq 0. \end{cases}$$

$\overrightarrow{f}$ is recovered by the inverse DCT as

$$f(n) = \sqrt{\frac{2}{N}} \sum_{u=0}^{N-1} c_u F(u) \cos\left(\frac{(2n+1)u\pi}{2N}\right).$$

The process of computing the inverse is referred to as *reconstruction*.

We can generalize the above to the d-dimensional DCT recursively. Let f be $N_1 \times N_2 \times \ldots \times N_d$ d-dimensional data, and F is the corresponding transformed coefficient set. The d-dimensional DCT for a coefficient $F_{\overrightarrow{u}}$ located at $(u_1, u_2, \ldots u_d)$ is defined as

$$F_{\overrightarrow{u}} = \sqrt{\frac{2^d}{\prod_{i=1}^{d} N_i}} \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} \cdots \sum_{m_d=0}^{N_d-1} \widehat{F},$$

where $\widehat{F} = f_{(m_1, m_2 \ldots, m_d)} \cdot \prod_{j=1}^{d} c_{u_j} \cos(\frac{(2m_j+1)u_j\pi}{2N_j})$, and the inverse DCT for the data point $f_{\overrightarrow{m}}$ located at $(m_1, m_2, \ldots m_d)$ is defined as

1. The average absolute relative error is defined as $\frac{1}{n}\sum_{i=1}^{n} \frac{|m_i - \widehat{m_i}|}{m_i}$, where m_i denotes the accurate value for the ith data point, and $\widehat{m_i}$ denotes the approximate value.

$$f_{\overrightarrow{m}} = \sqrt{\frac{2^d}{\prod_{i=1}^{d} N_i}} \sum_{u_1=0}^{N_1-1} \sum_{u_2=0}^{N_2-1} \cdots \sum_{u_d=0}^{N_d-1} \widehat{f},$$

where $\widehat{f} = \left[F_{(u_1,u_2\ldots,u_d)} \cdot \prod_{j=1}^{d} c_{m_j} \cos(\frac{(2m_j+1)u_j\pi}{2N_j}) \right]$.

2.2.1 Coefficient Selection

The number of DCT coefficients increases exponentially as the dimensionality increases. Clearly, computing all coefficients for possible selection is computationally prohibitive. Therefore, the DCT-based approaches choose and compute only the coefficients that are deemed the most representative. In practical OLAP systems, the data distribution should have a correlation among data items; that is, the frequency spectrum of the distribution should show large values in its low frequency coefficients and small values in its high frequency coefficients [8], [10]. In general, high-frequency coefficients are usually of less interest in OLAP, whereas low-frequency coefficients are of high interest, since they correspond to range-aggregation queries [20].

To select representative coefficients, several geometrical zonal sampling techniques such as triangular, spherical, rectangular, and reciprocal sampling have been proposed [21]. Lee et al. [10] extends the techniques to MDSs and shows that the reciprocal zonal sampling technique is able to select the most representative coefficients of DCT under the brown noise assumption. This sampling technique selects coefficients by the constraint $\{k_{\overrightarrow{u}} | \prod_{i=1}^{d}(u_i+1) \leq b\}$, where $k_{\overrightarrow{u}}$ is a DCT coefficient located at $(u_1, u_2, \ldots u_d)$, and u_i is the frequency index in the ith dimension.

Since only the most efficient coefficients need to be transformed, the execution time of DCT for data is reduced to $\mathrm{O}(nk)$, where k is the number of selected coefficients. Thus, the reciprocal zonal sampling technique improves the execution efficiency significantly.

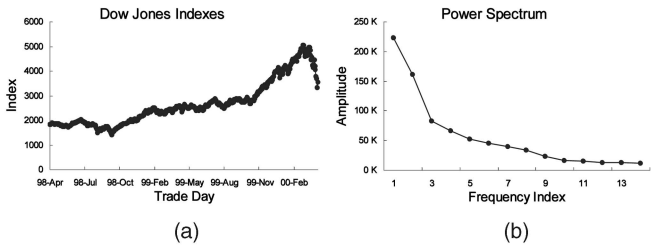
2.2.2 Answering Range-Sum Queries

For range-sum queries, it is expensive to estimate the value of individual data points and then compute the aggregation. Lee et al. [10] introduced the integral approach to compute the aggregation efficiently. For a d-dimensional range-sum query, it costs $\mathrm{O}(kd)$ to compute the aggregation if k coefficients are used. For example, the answer of the range-sum query requesting the sum of those cells located at $a < u_1 < b$ and $c < u_2 < d$ in an $M \times N$ 2D data cube can be estimated as $S = \int_c^d \int_a^b \widehat{f}(u_1, u_2) du_1 du_2,$, where $\widehat{f}$ is defined in Section 2.2.

Though not proper for estimating single data point [22], the DCT-based approaches are able to provide high-quality answers for range-sum queries due to the compensation of errors of some cell pairs. Because it does not require extra storage space for coefficient transforming, the DCT works well with small working space. However, the time complexity, estimated as $\mathrm{O}(nB)$, is deemed too high for the SCS applications to process the MDS blocks online.

3 The DAWA Algorithm

To resolve the issues of the space cost in DWT and the time cost in DCT, we propose the DAWA algorithm, which stands for an integrated algorithm of DCT for Data and DWT, to approximate the SCS. The DAWA algorithm is able to generate a snapshot from a PB in a small working buffer via one data scan, with a computational complexity of only $\mathrm{O}(\frac{nB}{\sqrt{N}} + B\log\sqrt{N})$, which we discuss further in Sections 3.2 and 3.3. The DAWA framework comprises two phases for generating snapshots. The first phase, called the *DCT phase*, partitions a PB into several subblocks, called DAWA cells, and then applies DCT to each DAWA cell based on the optimal set of coefficients selected by the reciprocal zonal sampling. The second phase, called the *DWT phase*, groups the DCT coefficients from each DAWA cell, whose time-to-frequency transformations are close to each other, into several isofrequency MDSs, denoted as IMS. As it will be shown in Section 3.1, the variance of data points in the same IMS is small, and the IMS can thus be further compressed by DWT efficiently. Also, several techniques, including null map, IMS smoothing, and global thresholding, have been developed based on the characteristics of IMS to improve the accuracy of the DAWA algorithm.



(a) (b)

Fig. 4. (a) The sequence for Dow Jones indices during 500 trading days. (b) The corresponding energy spectrum.

3.1 Brown Noise

The essence of DCT or DWT is based on the effectiveness of power concentration of the transformations. Basically, brown noise and random walks are the prevalent formats in real signals [23]. For brown noise, the energy is concentrated more in low frequencies, since the data points are more related to each other than those in white noise generated by the pure random process. Note that real signals have a skewed energy spectrum [8]. For example, stock movements and exchange rates can be successfully modeled as brown noise, which exhibit an energy spectrum $\mathrm{O}(f^{-2})$. Therefore, the DCT coefficients of such data, referred to as the amplitudes of the signals, can be modeled as $\mathrm{O}(f^{-1})$ [8]; that is, the amplitude is inversely proportional to the frequency. A scenario of the brown noise assumption for cube streams is shown in Fig. 4, where it can be seen that the amplitudes are inversely approximately proportional to the frequencies.

Also, the relationship between power density and frequencies can be explored mathematically to provide a theoretical foundation for zonal sampling. Consequently, we have the following lemma.

Lemma 1. *For a data series generated by a random walk process $\{X_n, 0 \leq n < N\}$ and the corresponding DCT coefficients $\{F(k), 0 \leq k < N\}$, the squared values of the coefficients $S(k)$, regarded as the spectral densities of $X(n)$, are reciprocally proportional to the squared frequency indices of the coefficients, that is, $S(k) \propto \frac{1}{k^2}$.*

Proof. Brown noise is a random process $X(t)$ with Gaussian increments, and

$$var(X(n_2) - X(n_1)) \propto |n_2 - n_1|.$$

Also, for a statistically self-similar data sequence $X(n)$ with the Hurst parameter H [24], the relationship between data points can be formulated as $var(X(n_2) - X(n_1)) \propto |n_2 - n_1|^{2H}$. It follows that

$$X(n_0 + n) - X(n_0) =_d \frac{1}{r^H}(X(n_o + rn) - X(n_0)) \quad (1)$$

for any n_0 and $r > 0$, where $=_d$ denotes equality in distribution.

The Hurst parameter for Brownian motion is $\frac{1}{2}$. For the case where $n_0 = 0$, and $X(n_0) = 0,$, the two random functions $X(n)$ and $\frac{1}{r^H}X(rn)$ are statistically indistinguishable.

For DCT, if the data sequence is very long, the transformation can be represented in a continuos form [10]. Thus, we reformulate DCT by the integral expression:

$$F(k, N) = \int_0^N \left(\cos\frac{(2n+1)\pi k}{2N}\right) dn$$

and the power spectral density of $X(k, T)$ is

$$S(k, N) = \frac{1}{N}|F(k, N)|^2.$$

The spectral density of X is then calculated as $N \to \infty$:

$$S(k) = \lim_{N\to\infty} \frac{1}{N}|F(k, N)|^2.$$

From (1), we can rewrite the original data series as a function of index and length:

$$Y(n, N) = \begin{cases} Y(n) = \frac{1}{r^H}X(rn) & \underline{\text{if } 0 < n < N} \\ 0 & \text{otherwise.} \end{cases}$$

By adopting the notations

$$\begin{array}{cc} Fx(n, N), F_Y(n, N) & \text{DCT of } X(n, N), Y(n, N), \\ Sx(n, N), S_Y(n, N) & \text{spectral densities of } X(n, N), Y(n, N), \\ Sx(k), S_Y(f) & \text{spectral densities of } X(n), Y(n), \end{array}$$

we have $F_Y(k, N) = \frac{1}{r^H}\int_0^N Y(n)\left(\cos\frac{(2n+1)\pi k}{2N}\right)dn$.
After substituting $\frac{2s+1}{2r} - \frac{1}{2}$ for n, and $\frac{ds}{r}$ for dn, we get

$$\begin{aligned} F_Y(k, N) &= \frac{1}{r^H}\int_0^{rN} X(s)\left(\cos\frac{(2s+1)\pi\frac{k}{r}}{2N}\right)\frac{ds}{r} \\ &= \frac{1}{r^{H+1}}F_X\left(\frac{k}{r}, rN\right). \end{aligned}$$

Now, it follows that for the spectral density of $Y(n, N)$

$$S_Y(k, N) = \frac{1}{r^{2H+1}}\frac{1}{rN}\left|F_X\left(\frac{k}{r}, rN\right)\right|^2,$$

and in the limit, as $rN \to \infty$, we conclude that

$$S_Y(k) = \frac{1}{r^{2H+1}}S_X\left(\frac{k}{r}\right) => S_X(1) = \frac{1}{r^{2H+1}}S_X\left(\frac{1}{r}\right).$$

By substituting k for $\frac{1}{r}$, we get

$$S_X(k) = \frac{1}{k^{2H+1}}S_X(1) \qquad \text{that is, } S_X(k) \propto \frac{1}{k^2}$$

for a brown noise. □

Lemma 1 shows the advantage over the information concentration of DCT. Also, note that the frequency of a coefficient depends on its location in the transformed PB. It is shown by Parseval's Theorem [25] that the energy of a signal conserves in both time and frequency domains. In addition, the Mean Squared Error (MSE) in the recovered data is the same as the MSE in the compressed coefficients. That is, the quality of recovered data reconstructed by the selected coefficients can be evaluated by the sum of the squared values of those coefficients. Thus, with a selected coefficient set $\mathcal{T} = \{k_{u_{11},\ldots,u_{1d}}, k_{u_{21},\ldots,u_{2d}}, \ldots, k_{u_{n1},\ldots,u_{nd}}\}$ in a d-dimensional data cube, the quality of $\mathcal{T}$ can be defined as $H(\mathcal{T}, n) = \sum_{i=0}^{n-1}(k_{u_{i1},\ldots,u_{id}})^2$, where n is the size of $\mathcal{T}$, and u_{ij} is the index of the ith coefficient for dimension j.

Consequently, we formally prove by the following lemma that based on the assumption of brown noise distribution, reciprocal zonal sampling is, in fact, the most efficient method for coefficient selection.

Lemma 2. *For a symmetric d-dimensional data cube with brown noise distribution, the selection of coefficients with the best quality is $T = \{k_{p_i} | \prod_{j=1}^{d}(u_{ij} + 1) \le b\}$, where b is a constant.*

Proof. Without loss of generality, we assume that the data cube is 2D. Therefore, the DCT coefficients can be modeled as $k_{(m,n)} = \frac{K}{(m+1)(n+1)}$, where K is a positive constant, and m and n represent the frequencies of the x-axis and y-axis, respectively. Given that

$$k_{(x,y)} \in T = \{k_{(m,n)} | (m+1)(n+1) \le b\},$$

it follows that $(x+1)(y+1) = b$ holds for the smallest coefficient. If there exists another selection of coefficients T', which causes $H(T', n) > H(T, n)$, there must exist a coefficient $k_{(x,y')} \in T'$, where $\frac{K}{(x+1)(y'+1)} > \frac{K}{(x+1)(y+1)}$. This implies that $(x+1)(y'+1) < b$, which is a contradiction. Therefore, the lemma follows. □

Moreover, the statement of Lemma 2 can be extended to the case of asymmetric dimensions under the brown noise assumption of data distribution. In the case of an asymmetric data cube, we can multiply the index for each dimension of a DCT coefficient by a scaling factor in order to make all the indices proportional to the corresponding frequencies. The relationship between the frequency and cell index can be expressed as $f_{ij} = \frac{u_{ij}+1}{|D_j|}$. Asymmetric reciprocal zonal sampling is defined as follows:

Corollary 2.1. *For an asymmetric d-dimensional data cube with brown noise distribution, the selection of coefficients with the best quality of the data cube is $\{k_{p_i} | \prod_{j=1}^{d}(u_{ij} + 1) \le b'\}$, where b' is a constant.*

Proof. From the definition of asymmetric cubes and Lemma 2, the selection of coefficients with the best quality can be defined as $T = \{k_{p_i} | \prod_{j=1}^{d}(\frac{u_{ij}+1}{|D_j|}) \le b\}$, where $|D_j|$ is the size of dimension D_j. Since $\prod_{j=1}^{d}|D_j|$ is a constant for a given cube, T can be rewritten as

TABLE 2
Notations of the DAWA Algorithm

Notation	Meaning
N	Cardinality of a PB
n	Number of non-zero data points
k	Number of DAWA cells
M^p	A data point located at p
$X(M^p)$	Measure of M^p
D^q	A DAWA cell located at q
E_r^q	A data point or coefficient located at r in D^q
I_u	The IMS collected from coefficients at u
W^u	Result of decomposing I_u by DWT
W_v^u	The DWT coefficient located at v in I_u

$$T = \left\{ k_{p_i} | \prod_{j=1}^{d} (u_{ij}+1) \le b \cdot \prod_{j=1}^{d} |D_j| \right\} = \left\{ k_{p_i} | \prod_{j=1}^{d} (u_{ij}+1) \le b' \right\}.$$

From Lemma 2, it is shown that the reciprocal zonal sampling is optimal for information conservation.

3.2 The DCT Phase

In the DCT phase, both the PB and the working buffer are first partitioned into k subblocks and then perform DCT to each subblock of the PB, called the DAWA cell, to obtain $\frac{B}{k}$ coefficients with the reciprocal zonal sampling. The transformed coefficients of DCT are stored in one subblock, called T-DAWA cell, in the working buffer. In other words, a DAWA cell is transformed by DCT and is stored in the corresponding T-DAWA cell. The flow of the *DCT phase* is given as follows:

1. Partition PB into k DAWA cells.
2. Find a multidimensional data point and translate its coordinate, which corresponds to the PB, into that corresponding to DAWA cell.
3. Perform DCT on the incoming data point and store the $\frac{B}{k}$ coefficients in the corresponding T-DAWA cell.
4. Repeat steps 2 and 3 until all data points in the PB have been processed.

Note that the DCTs over the DAWA cells are symmetric, thus avoiding the problem of dimension order. Also, the number of resultant coefficients is equal to the size of the working buffer. For ease of presentation, the symbols used in the following sections are summarized in Table 2.

The transformation T_W^D is defined to transform the coordinate p of the data point M^p in the PB to a new coordinate $\{q, r\}$ of the data point E_r^q in the DAWA cell D^q. The concept of the transformation of the PB into 18 subblocks is shown in Fig. 5. For the original data point located at $(\mathrm{X}, \mathrm{Y}, \mathrm{Z}) = (8, 1, 0)$ in the PB, the corresponding DAWA cell is $\mathrm{D}^{(2,0,0)}$, and the coordinate of E_r^q is $(2, 1, 0)$. After performing T_W^D, each data point is related to one DAWA cell only. Since only the contribution of a data point to the corresponding T-DAWA cell, rather than its contribution to the entire working buffer, needs to be calculated, the execution efficiency is improved significantly.

Intuitively, the time for performing DCT on k DAWA cells is $1/k$ of that for performing DCT on the entire PB with the same working buffer. However, partitioning too many DAWA cells will degrade the efficiency of the *DWT phase*. The more DAWA cells a PB has, the larger the sequence that the Haar wavelets have to decompose in this phase. In an extreme case, where the number of DAWA cells is the same as the number of data points of the PB, we have a pure DWT-based approach. On the other hand, the process becomes a DCT-based approach if the volume of a DAWA cell is the same as that of the PB. The strategy for optimizing this trade-off is to partition each dimension to an appropriate size that is close to the square root of the dimension size, which reduces the computational complexity from $\mathrm{O}(nB)$ to $\mathrm{O}(\frac{nB}{\sqrt{N}})$, by transforming a PB to B coefficients. In general, the value of N is large; thus, the execution efficiency is improved significantly.

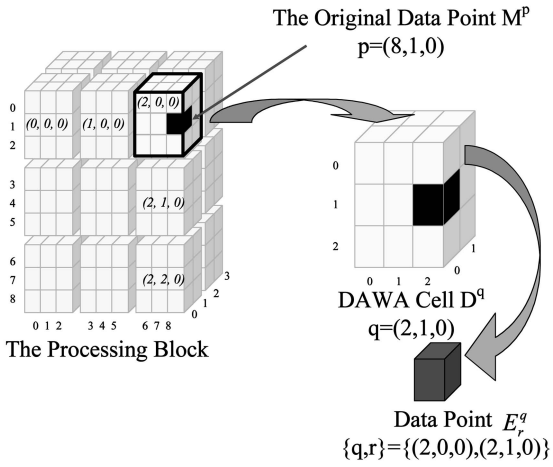


Fig. 5. The concept of $\mathrm{T}_W^D : (8, 1, 0) \rightarrow \{(2, 0, 0), (2, 1, 0)\}$.

In addition to reducing the overall computational complexity of performing DCT, the effectiveness of the reciprocal zonal sampling is another factor that affects the quality of compression. To show that the efficiency is not compromised by partitioning a PB into DAWA cells, we now discuss the data distribution of data series in a DAWA cell.

Lemma 3. *For a data series generated by the random walk process* $\{X_t, 0 \le t < T\}$, *the subset of* X_t, $\{X_{t'}, 0 < t_1 \le t' \le t_2 < T\}$, *is also a brown noise.*

Proof. For a data series generated by the random walk process, it can be written as

$$X_t = X_{t-1} + Z_t = X_0 + \sum_{i=1}^{t} Z_i.$$

Since Z_t is a purely random process, $X_{t'}$ can be rewritten as $X_{t'} = X_0 + \sum_{i=1}^{t'} Z_i = (X_0 + \sum_{i=1}^{t_1} Z_i) + \sum_{i=t_1+1}^{t'} Z_i$.

Without loss of generality, we denote $(X_0 + \sum_{i=1}^{t_1} Z_i)$ as $X_0^{'}$, with some manipulation. The reindexed series can then be formulated as $X_{t''} = X_0^{'} + \sum_{j=0}^{t'-t_1} Z_j^{'}$. Since $\{Z_j^{'}\}$ is also a white noise, the subset $\{X_{t'}, 0 < t_1 \le t' \le t_2 < T\}$ is thus a brown noise. □

From Lemma 3, the data series along each dimension in a DAWA cell is shown to be a brown noise, so the power density can be estimated as $S(k) \propto \frac{1}{k^2}$ according to Lemma 1. Therefore, performing the reciprocal zonal sampling on a

single DAWA cell is as effective as performing it on the whole PB from a local perspective; that is, the local optimal selection of coefficients can be achieved. For global optimization, the correlations between the DAWA cells can be further utilized by performing DWT on the DCT coefficients located in different T-DAWA cells.

3.3 The DWT Phase

The task of the *DWT phase* is to collect coefficient sets from different T-DAWA cells to be further compressed by DWT in order to take the advantage of the similarity between the distributions of each T-DAWA cell. Intuitively, the smaller the degree of entropy that this set has, the higher the compression rate that DWT can achieve. Consequently, DWT prefers coefficients whose absolute values are close to each other. In light of this, DCT coefficients located at the same coordinate u in each T-DAWA cell are collected as an IMS I_u to be decomposed by DWT. In a T-DAWA cell, the absolute value of a DCT coefficient is inversely proportional to $\prod(u_i+1)$, where u_i is the frequency index in the ith dimension. For example, the coefficient $E^r_{(0,1)}$ is expected to be twice as large as $E^r_{(1,1)}$ in magnitude.[2] Here, the coefficient whose index of each dimension is equal to 0 is called the DC coefficient, and the corresponding IMS is denoted as I_0.[3] For each T-DAWA cell D^i, if all coefficients in D^i are adjusted by a scaling factor, where the scaling factor is the ratio of the DC coefficient of D^0 to that of D^i, the collected IMS will be much smoother. This technique, called scaling, converts a T-DAWA cell to a scaled T-DAWA cell and smoothes the magnitude of the coefficients. Note that these factors can be calculated from I_0; thus, no extra storage space is needed. As a result, the decomposed IMS, called D-IMS, has the same structure as that of an IMS, but it is much more representative of the original information.

The processes of the DWT phase are given as follows: 1) For each T-DAWA cell D^i, multiply all the coefficients, except the DC coefficient of D^i, by the corresponding scaling factor, 2) collect the IMSs from the T-DAWA cells and perform DWT on each of them, and 3) sort all the decomposed coefficients in each D-IMS, perform the heuristics, and select the coefficients to store in the synopsis memory. Since the coefficients in an IMS are collected from each T-DAWA cell, the working buffer B, which is partitioned into k parts, is able to accommodate $\frac{B}{k}$ D-IMSs. The process of collecting the IMSs from the T-DAWA cells is illustrated in Fig. 6. It shows that the data points in the middle right of each T-DAWA cell are collected to an IMS.

To prove the optimality of IMSs for wavelets decomposition, the relationship between an IMS and a reconstructed IMS needs to be explored further. For brevity, we employ the case of a four-point series to illustrate the nature of an IMS.

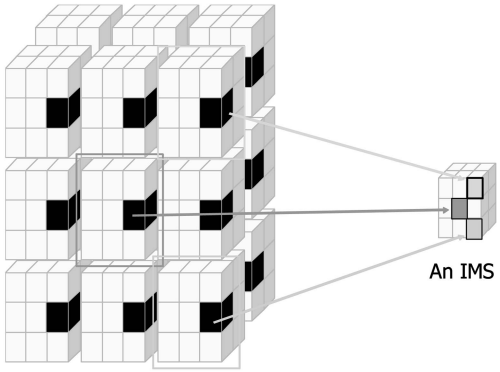


Fig. 6. The process of collecting the IMSs from the T-DAWA cells.

Lemma 4. *For two four-point IMSs of* $\mathrm{IMS}_1=\{X^1_0,X^1_1,X^1_2,X^1_3\}$ *and* $\mathrm{IMS}_2=\{X^2_0,X^2_1,X^2_2,X^2_3\}$, *the* L_2 *error*[4] *of DWT will increase if* X^1_j *and* X^2_j *are exchanged, and the hard-threshold pruning drops the coefficient with the minimum magnitude.*

Proof. By performing DWT, we can estimate the decomposed coefficients as

$$W^i=\left\{\frac{X^i_0+X^i_1+X^i_2+X^i_3}{2},\frac{X^i_0+X^i_1-X^i_2-X^i_3}{2},\frac{X^i_0-X^i_1}{\sqrt{2}},\right.$$
$$\left.\text{and }\frac{X^i_2-X^i_3}{\sqrt{2}}\right\},$$

where $i=1,2$. In [16], it is shown that pruning the coefficient with the smallest magnitude achieves the best reconstruction result under the thresholding scheme. Also, Garofalakis and Gibbons [26] show that the expected value of the overall L_2 error of reconstructing the IMS, denoted as $E[L_2]$, is $\sum_{i|c_i} Var(c_i)$, where c_i is the normalized coefficient. Thus, the efficiency of DWT with one-coefficient pruning can be evaluated by the squared value of the smallest decomposed coefficient.

Let W^i_j be the jth item of W^i and e^i_j be the error estimator corresponding to W^i_j. By the definition of IMS proposed in Section 3, the absolute values of X^i_0, X^i_1, X^i_2, and X^i_3 should be close to each other, and the constraint $\frac{|X^i_n|}{|X^j_n|}=\frac{j}{i}$ should be held. Without loss of generality, the two IMSs can be rewritten as

$$I_1 = \{l^1_0(C+\Delta_1),l^1_1(C+\Delta_2),l^1_C C,l^1_3(C+\Delta_3)\},$$
$$I_2 = \left\{\frac{l^2_0(C+\Delta_1)}{r},\frac{l^2_1(C+\Delta_2)}{r},\frac{l^2_C C}{r},\frac{l^2_3(C+\Delta_3)}{r}\right\},$$

where $l^i_j\in\{-1,1\}$, $\Delta_i\ll C$, and $r>1$.

If these two coefficients $l^1_3(C+\Delta_3)$ and $\frac{l^1_3(C+\Delta_3)}{r}$ are exchanged, we have

$$I_3 = \left\{l^1_0(C+\Delta_1),l^1_1(C+\Delta_2),l^1_C C,\frac{l^2_3(C+\Delta_3)}{r}\right\},$$
$$I_4 = \left\{\frac{l^2_0(C+\Delta_1)}{r},\frac{l^2_1(C+\Delta_2)}{r},\frac{l^2_C C}{r},l^1_3(C+\Delta_3)\right\}.$$

2. The ratio of expected values of $E^r_{(0,1)}$ to $E^r_{(1,1)}$ are $\frac{1}{(0+1)\cdot(1+1)}/\frac{1}{(1+1)\cdot(1+1)}=2$.

3. The value of the DC coefficient is equal to the product of the average of all data points in the same DAWA cell and a constant $\sqrt{\frac{2^d}{\prod_{i=0}^{d-1}|d_i|}}$.

4. The L_2-error is defined as $\frac{1}{n}\sum_{i=1}^{n}|m_i-\widehat{m_i}|^2$., where m_i denotes the accurate value for the ith data point, and $\widehat{m_i}$ denotes the approximate value.

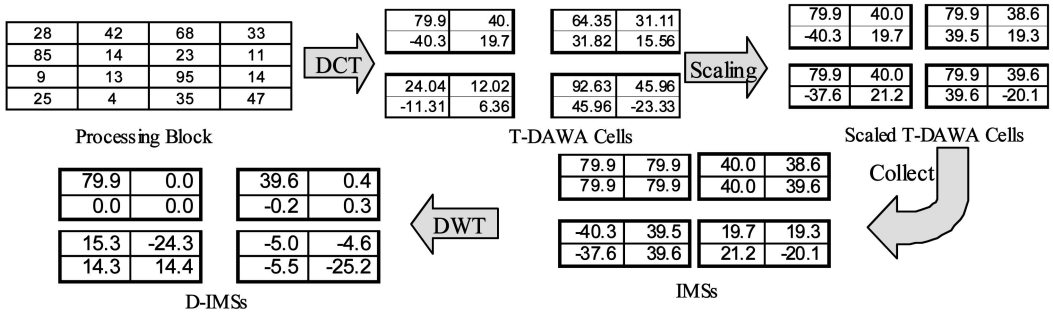


Fig. 7. The process for transforming a PB to D-IMSs.

Also, Err_D represents the overall L_2 error of performing DWT on I_1 and I_2 by dropping one coefficient for each D-IMS, and Err_x denotes the errors of performing DWT on I_3 and I_4. To prove the optimality of IMSs selected by the DAWA algorithm, it must be shown that Err_D is smaller than Err_x.

For the case of $l_1^i = -l_0^i$, the best (smallest) error would be e_2^i. However, since it is irrelevant to the analysis of exchanging coefficients, this case is ignored. For other cases, since all the absolute values of the coefficients of I_1 are close to C, it follows that there should be at least one W_i^1 whose absolute value is $\mathrm{O}(\Delta_j)$. For the cases of $l_1^i = l_0^i$, or $l_C^i = l_3^i$, it follows that $W_2^1 = 0$, or $W_3^1 = 0$. For the case of $l_1^i \neq l_0^i$, and $l_C^i \neq l_3^i$, either $W_0^1 = 0$ or $W_1^1 = 0$ holds. Thus, the minimum errors of W^1 and W^2 can be estimated as $O(\Delta^2)$ and $O(\frac{\Delta^2}{r})$ respectively, that is, $\mathrm{Err}_D \sim O(\Delta^2)$.

On the other hand, at least one of the W_i^3s will be $O(C - \frac{C}{r})$ if $l_1^1 \neq l_0^1$. As mentioned before, either $l_1^1 \neq l_0^1$ or $l_1^2 \neq l_0^2$ must be true, so we have $\mathrm{Err}_x \sim O((C - \frac{C}{r})^2)$.

Since the L_2 error will not improve after exchanging two coefficients of the IMSs, the selection of coefficients in IMSs results in an optimal L_2 error for wavelets decomposition. □

From Lemma 4, the IMS is shown to be the optimal selection for performing DWT. Since all the data sources for the DWT phase, which are generated from the DCT phase, are already in the working buffer, the DWT phase can be processed without performing thresholding during multidimensional decomposition. Therefore, the optimality of performing thresholding can be maximized.

Fig. 7 shows an example of the process for transforming a 4×4 PB to a D-IMSs, where the PB is partitioned into four DAWA cells. The four T-DAWA cells are first scaled, and the coefficients at the same coordinates in each scaled T-DAWA cell are collected to construct an IMS.

For the DWT-based approaches, it costs $O(k)$ to decompose a data sequence of size k. Thus, the computational complexity of the DWT phase can be deduced as $\mathrm{O}(\frac{B}{k} \cdot k) = \mathrm{O}(B)$. To store a large number of PBs in the synopsis memory, the size of each D-IMS must be as small as possible. Based on the theory of thresholding in DWT, only the most representative coefficients of D-IMSs are selected as the snapshot and stored in the synopsis memory. Instead of selecting the DWT coefficient set that is good for each individual IMS, three techniques, called null map, IMS smoothing, and global thresholding, have been developed to improve the overall accuracy of the DAWA algorithm.

3.3.1 Null Map

The null map is a heuristic that keeps track of empty DAWA cells. Since cube streams in several real applications are very sparse [18], [19], many of the DAWA cells are usually empty. Consequently, the corresponding T-DAWA cells will also be empty. Intuitively, empty DAWA cells do not contribute to the answers of range-sum queries. If the queries involve data points located in empty DAWA cells, the answers contributed by those data points should be estimated as zeros in order to improve both the execution efficiency and the answer quality. Therefore, the *null map* selectively skips the calculation of answers. It is noted that only one null map is needed for one PB, since all IMSs in the same coordinate have zero values with respect to empty T-DAWA cells.

For $[\frac{B}{k}]$ D-IMSs, that is, k T-DAWA cells, stored in the working buffer B, it costs $O(k \cdot e)$ storage space to keep track of those empty cells, where e is the probability of empty T-DAWA cells. The parameter e would be small, whereas the cube stream is sparse. In the worst case $(e = 1)$, the space cost of the null map is equal to that of one D-IMS. For a range-sum query whose selectivity[5] is s, the null map is able to reduce $O(s \cdot k)$ number of times to perform reconstruction of DWT and inverse DCT for answering this query.

3.3.2 IMS Smoothing

The DWT applications require more coefficients for rugged data series than for smooth data series to achieve the same reconstruction quality. As mentioned in Section 3.3.1, the sparseness of data sets makes the IMSs rugged. In light of this, those data points tracked by the null map in each IMS can be replaced with new values to smooth the corresponding IMSs. Hence, we employ a heuristic, called *IMS smoothing*, to replace the zeros in IMSs with appropriate values in order to reduce the entropy of IMSs. Thus, the quality of reconstruction can be improved by using the same compression rate. The new values of the original zero-value data points can be calculated from the average of the neighborhoods. Note that the smoothing technique is not beneficial for the traditional DWT applications, since the

5. The selectivity of a range-sum query is defined as the ratio of the number of cells involved in the query to the total number of cells in the cube.

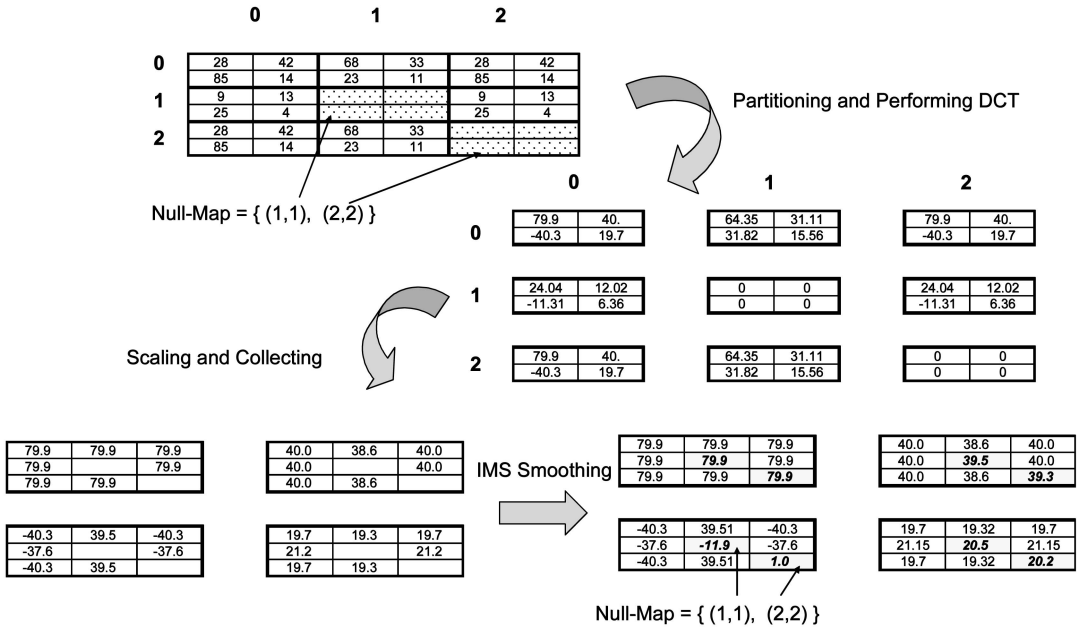


Fig. 8. The processes of null map and IMS smoothing.

storage space required to keep the coordinates of the replaced data points for a single data series could be larger than that for keeping the original information. Instead, the data points with zero values of different IMSs can be tracked from the same null map so that the storage space required to smooth IMSs is very low. For d-dimensional PB with k T-DAWA cells stored in the working buffer B, calculating the average for each empty cells in a D-IMS costs $O(d)$ in time complexity. Thus, the total time cost of IMS smoothing can be estimated as $O([\frac{B}{k}]\ d)$, showing that the time cost for performing the IMS smoothing is relatively small compared to the processes of the DCT phase and DWT phase.

The concepts of null map and IMS smoothing can be best understood by the following example.

Example 4. Fig. 8 shows the concepts of null map and IMS smoothing. The 6×6 PB is partitioned into nine DAWA cells, and the coordinates of the empty DAWA cells {(1, 1), (2, 2)} are added in the null map. After performing DCT, scaling, and collecting, four IMSs have been generated, and each of them has two empty cells. After performing the IMS smoothing, those empty cells are replaced by the averages of their neighborhoods. It is noted that only one null map is needed to track the empty cells in the original PB and those IMSs.

3.3.3 Global Thresholding

The third heuristic that is used to improve the performance of the *DWT phase* performs hard thresholding on all the coefficients in the working buffer together rather than performing it for each D-IMS individually. Assuming that the size of the available synopsis memory for a PB is limited to b, one should select the most b representative coefficients. The *DWT phase* adopts global thresholding to select the top-b coefficients from all D-IMSs. Note that D-IMSs with too few coefficients after global thresholding will degrade the quality significantly. Thus, the threshold, denoted as T_{noc}, is set to filter out low-quality D-IMSs.

The process of global thresholding is given as follows:

1. Sort the coefficients of the D-IMSs.
2. Mark the top-b coefficients.
3. Drop those D-IMSs that do not meet the constraints imposed by T_{noc} and unmark the corresponding coefficients.
4. Mark more coefficients that have large absolute values from the remaining D-IMSs until the number of marked coefficients reaches the memory limit b.

The marked coefficients are then stored in the synopsis memory.

For $[\frac{B}{k}]$ D-IMSs, that is, k T-DAWA cells, stored in the working buffer B, sorting for each D-IMS costs $\mathrm{O}(k \log k)$ in time complexity, whereas the step that merges $[\frac{B}{k}]$ sorted series to mark the top-b coefficients is of the complexity $\mathrm{O}(k \cdot [\frac{B}{k}])$. Therefore, the computational complexity for the global thresholding is $O([\frac{B}{k}] \cdot k \cdot \log k + k \cdot [\frac{B}{k}])$, and that of algorithm DAWA, which can be analyzed from the operations on the DCT phase, and the DWT phase and heuristics is $\mathrm{O}(\frac{nB}{\sqrt{N}} + B \log \sqrt{N})$.

3.4 Answering Range-Sum Queries

For the range-sum queries over a PB, the answers can be estimated in the inverse order of the DCT and DWT phases. The steps of this process are 1) estimate the range of involved DAWA cells, 2) reconstruct the coefficients of IMSs related to T-DAWA cells, and 3) apply the integral techniques to estimate the contribution of each DAWA cell and return the summed results.

As mentioned before, the null map ignores the effect of empty DAWA cells when answering queries. For fully involved DAWA cells, that is, DAWA cells that are totally involved in a range-sum query, their contributions can be estimated directly from the values of the DC coefficients.

Algorithm DAWA
Input:(d, B, b, $\{M^p\}$, T_{noc}}, d: # of dimensions;
{M^p}: a PB;{N$_i$}:sizes of dimensions
Output: A set of coefficients H and a NullMap M
1. Partition the PB to $\prod_{i=1}^{d}\lceil\sqrt{N_i}\rceil$ sub-blocks
2. Set a NullMap M, size of i-th dimension=$\sqrt{N_i}$
3. For each M^p
{ $(D^q, E_r^q) = \mathrm{T}_W^D(M^p)$
Perform DCT on each E_r^q in the DAWA cell D^q
Update I_q ; Set $M(q) = 1$ }
4. Scaling all T-DAWA cells
5. For each IMS I_i
{Perform DWT on I_i to a D-IMS W^i;
Sort W_j^i } //W_j^i is the j-th coefficient of W^i
6. Merge-sort all W^i, mark b largest coefficients
7 For each W^i smaller than T_{noc} and T_{eng}
{ u=number of marked coefficients in W^i;
Drop W^i *and* mark u more coefficients from remaining D-IMSs}
8. Set H = union of all the marked W_j^i
9. return H, M

Fig. 9. The DAWA algorithm.

Intuitively, the I_0, which keeps the DC coefficient for each T-DAWA cell, should be kept accurately.

By performing the efficient partitioning technique and applying DCT in the *DCT phase*, the DAWA algorithm transforms the original PBs into highly representative coefficients efficiently. In the *DWT phase*, the transformed DCT coefficients are organized into IMSs, which has been shown to be the most efficient method for decomposition. DWT is then applied to each IMS to obtain more representative coefficients. Finally, the null map, IMS smoothing, and global thresholding are applied to select the best coefficient set as the snapshot to be retained in the synopsis memory. For range-sum queries, the mechanisms integrating DCT and DWT are proposed to return answers quickly. Moreover, the heuristics for fully involved and empty DAWA cells enable the DAWA algorithm to achieve better execution efficiency, resulting in answers of good quality. An algorithmic form of DAWA is given in Fig. 9.

4 Experimental Results

To evaluate the DAWA algorithm, we conducted three sets of empirical studies on both synthetic and real data sets. In Section 4.1, the scalability of the DAWA algorithm with various settings of working buffer and synopsis memory for a PB is evaluated. In Section 4.2, DAWA's performance generating snapshots of PBs is compared to that of the wavelet-based approach [11]. The quality of the answers of range-sum queries for both DAWA and the wavelet-based approach is then evaluated in Section 4.3.

The synthetic data set, denoted as D-TPC, which is taken from the decision support benchmark, TPC-H [27], consists

TABLE 3
Sizes of PBs and DAWA Cells

	D-TPC	D-TEL
Size of a PB	3×48×5×25×25×5	24×58×8×113
Size of a DAWA Cell	2×7×3×5×5×3	6×9×2×10
Number of DAWA Cells	2×7×2×5×5×2	4×7×4×12

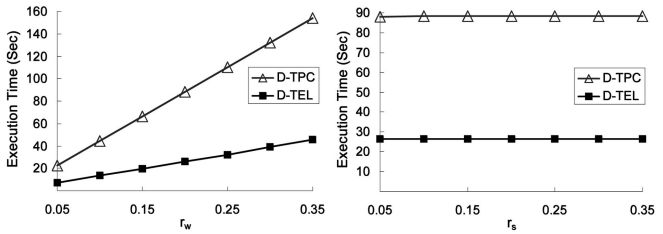


Fig. 10. Execution times for DAWA under (a) a constant $r_s = 0.1$ and various values of r_w and (b) a constant $r_w = 0.2$ and various values of r_s.

of eight relations. Since the SCS applications on MDSs, a derived table consisting of six dimensions (*Suppliers*, *Nationkey*, *Orderstatus*, *Mkesegment*, *Mfgr*, and *Brand*) is joined by six relations to compose a cube with 169,665 tuples and 2,250,000 cells. Also, a real-world cube stream, denoted as D-TEL, is obtained from the error log of a large cellular phone company. Four common dimensions are considered, where 1,258,368 cells and 120,478 tuples are involved in this cube. The sizes of the PBs and the DAWA cells for both data sets are listed in Table 3.

To simulate range-sum queries, the range in each dimension of a query is randomly decided based on the criterion that the product of all ranges is close to the query selectivity 0.1. Also, the working buffer size is controlled by the parameter r_w, which is defined as $r_w = \frac{B}{N}$. The size of the available synopsis memory is controlled by $r_s = \frac{b}{B}$.

4.1 Scalability of DAWA

To verify the scalability of the DAWA algorithm, both D-TPC and D-TEL are compressed by DAWA, with various settings of r_w and r_s. The execution times for generating the snapshots for the PBs are shown in Fig. 10.

It can be seen in Fig. 10 that the execution time of DAWA is proportional to the size of the working buffer and that the performance is not degraded as the size of the snapshots increases. This shows that the inclusion of the heuristics proposed in Section 3 does not affect the execution efficiency and that the DAWA algorithm scales well with different sized snapshots.

To evaluate the sensitivity of error relative to r_w, the average absolute relative error (1-norm error) in answering 100 queries by DAWA for both $\mathrm{D_{TEL}}$ and $\mathrm{D_{TPC}}$ are measured with various settings of r_w. It can be seen in Fig. 11 that the error rate decreases as the size of the working buffer increases, showing that a compromise between execution time and accuracy can be achieved.

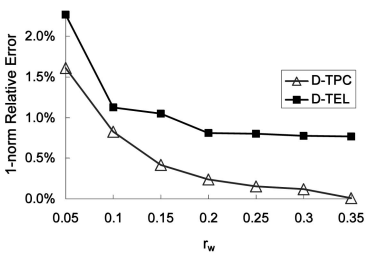


Fig. 11. Absolute relative (1-norm) error rates with, various settings of r_w.

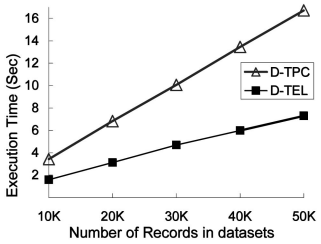


Fig. 12. Execution times of different sizes of data sets.

To show the good scalability of DAWA in dealing with different sizes of data sets, five small data sets of different sizes are sampled from both D-TPC and D-TEL. Fig. 12 shows the linear relationship between the execution time and the size of data set. That is, the execution time is proportional to the number of tuples of the original data set. This conforms to the computational complexity $O(Bn)$, where B is the size of the working buffer, and n is the number of nonnull tuples in the original data set.

Note that the r_w and r_s in this experiment are set over a wide range in order to evaluate the scalability of DAWA. The following experiments are performed with small r_w and r_s to evaluate the performance of the DAWA algorithm in severely constrained environments.

4.2 Performance of Snapshot Generation

To assess the performance of the DAWA algorithm, the wavelet-based approach with a hard-thresholding mechanism [11], abbreviated as DWT, is implemented for comparison purposes. The execution times for generating snapshots of both the DAWA algorithm and DWT are measured for D-TPC and D-TEL, with various settings of r_s. To compare the performance of both algorithms fairly, the I/O times for scan data are excluded, but the time cost of decomposition, transformation, and heuristics is considered. As shown in Fig. 10a, the execution time for the DAWA algorithm is almost linear to the size of the working buffer. Therefore, r_w is set to a constant (0.05), and the execution time for different sizes of the working buffer can be estimated proportionally.

Fig. 13 indicates that the DAWA algorithm outperforms the DWT algorithm in execution efficiency by a margin of 3.5 times for the D-TEL data set. Furthermore, in a large D-TPC data set, the DAWA algorithm is eight times more efficient than the DWT algorithm. Consequently, the computational complexity of DWT, estimated as $O(N)$ in Section 2, is much larger than that of the DAWA algorithm.

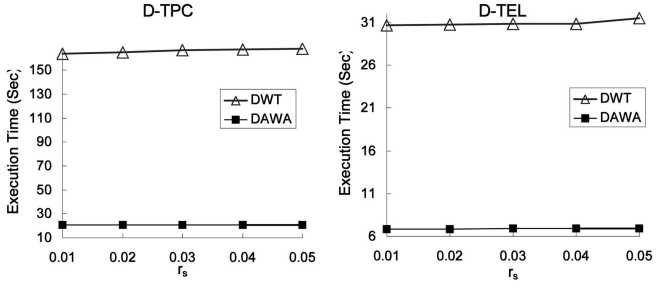


Fig. 13. Execution times of generating snapshots, with various values of r_s.

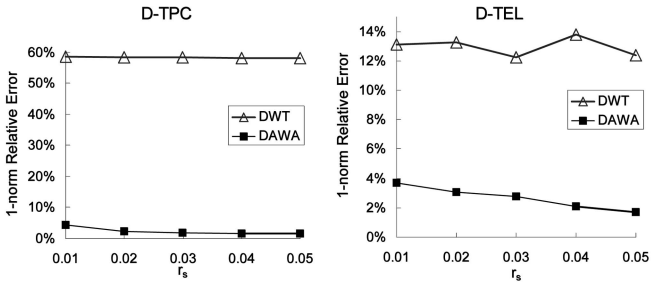


Fig. 14. Relative (1-norm) error rates of answers, with various settings of r_s.

Note that the execution times for the DCT-based approaches [10] are not included, since the computational complexity is too high to be used for the SCS applications. For the case of $r_s = 0.01$, the DCT-based approaches take more than 8,500 seconds to finish the experiment, which means that the time cost is 1,400 times greater than that of DAWA.

4.3 Quality of Answers

To evaluate the quality of answers for range-sum queries, 100 queries for both D_{TEL} and D_{TPC} are generated. The settings of r_w and r_s are the same as those in Section 4.2.

As shown in Fig. 14, the quality of answers from the DAWA algorithm is much better than that from DWT. The error rates of the answers from DAWA are all below 5 percent for both D-TPC and D-TEL, showing that DAWA works well under a small working buffer and is able to generate very small snapshots for the original data sets, with acceptable error rates. In contrast, the DWT error rates are as high as 14 percent and 58 percent for D-TEL and D-TPC, respectively, because the working buffer is insufficient. Furthermore, DWT needs to perform hard thresholding and drops coefficients after the decomposition of each dimension so that its thresholding mechanism degrades the quality of answers significantly. Moreover, the quality of answers may not improve if more storage space for the synopsis memory is supplied.

4.4 Experimental Studies of the Heuristics

To evaluate the improvement in the quality of answers and the execution performance contributed by null map and IMS smoothing, the range-sum queries in Section 4.3 for both DTEL and DTPC are used. The quality of answers are evaluated in three different cases. The first is by using the original DAWA algorithm while both null map and IMS smoothing are active. The second one, denoted as NoNullMap, deactivates the null map, whereas the third, denoted as NoIMS, uses the DAWA algorithm without performing the IMS smoothing. Since the IMS smoothing is performed during the DWT phase, there will be no difference between the DAWA algorithm and NoIMS in the execution performance of answering queries. Thus, only the execution performance of the DAWA algorithm and NoNullMap are compared. The performance of the three settings are measured for D-TPC and D-TEL, with various settings of r_w and a fixed setting of $r_s = 0.1$.

Fig. 15 indicates that the execution performance of the DAWA algorithm is improved while the null map is activating. It can be seen that the percentages of the reduced execution time from different settings of r_w are close to each

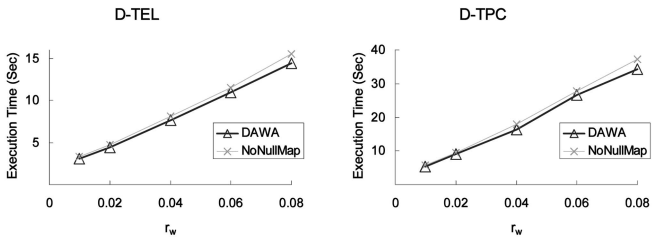


Fig. 15. Execution times of generating snapshots, with various values of r_w.

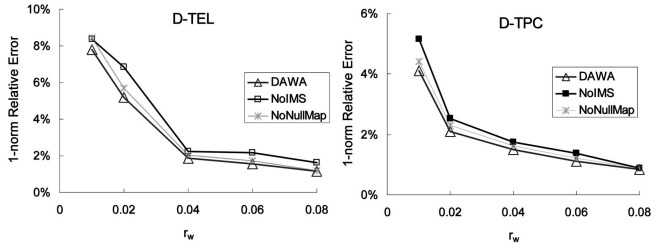


Fig. 16. Relative (1-norm) error rates of answers, with various values of r_w.

other, showing that the execution times reduced by performing null map is proportional to the number of DAWA cells and independent of the size of working buffer.

As shown in Fig. 16, the quality of answers from the DAWA algorithm is better than that from both NoIMS and NoNullMap. The error rates reduced by those two heuristics vary in the range from 0.5 percent to 1 percent; that is, the quality of answers are improved by a margin of 3 percent to 30 percent. Also, Fig. 16 shows that the improvement from the IMS smoothing is slightly better than that from null map. It is noted that the quality of answers of all those three settings are better than that of DWT in Section 4.3.

5 Conclusions

In this paper, we have addressed the problem of generating snapshots for cube streams and proposed an efficient approach, called DAWA, for PBs. By utilizing the techniques of partitioning and selecting IMSs, DAWA is able to compress a PB into highly representative coefficients efficiently during the DCT phase and the DWT phase. In addition, three heuristics have been developed to generate the final results for storage in the synopsis memory and improve the quality of answers. As shown by our experimental results, as it combines the merits of DCT and DWT, DAWA not only generates snapshots for PBs very rapidly but also delivers high-quality answers for range-sum queries.

Acknowledgments

This work was supported in part by the National Science Council of Taiwan Contract NSC93-2752-E-002-006-PAE.

References

[1] E. Codd, S. Codd, and C. Salley, "Providing OLAP (On-Line Analytical Processing) to User-Analysis: An IT Mandate," technical report, Arbor Software Corp., 1993.

[2] W.-G. Teng, M.-S. Chen, and P.S. Yu, "Using Wavelet-Based Resource-Aware Mining to Explore Temporal and Support Count Granularities in Data Streams," *Proc. Fourth SIAM Int'l Conf. Data Mining (SDM '04)*, 2004.

[3] D. Zhang, D. Gunopulos, V.J. Tsotras, and B. Seeger, "Temporal Aggregation over Data Streams Using Multiple Granularities," *Proc. Int'l Conf. Extending Database Technology*, pp. 646-663, 2002.

[4] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining Stream Statisitcs over Sliding Windows," *Proc. 13th ACM-SIAM Ann. Symp. Discrete Algorithms (SODA '02)*, 2002.

[5] M.-Y.Y. Bi-Ru Dai, J.-W. Huang, and M.-S. Chen, "Adaptive Clustering for Multiple Evolving Streams," *IEEE Trans. Knowledge and Data Eng.*, vol. 18, no. 9, p. 1166, Sept. 2006.

[6] H.-P. Hung and M.-S. Chen, "Efficient Range-Constrained Similarity Search from Wavelet Synopses over Multiple Streams," *Proc. 15th ACM Conf. Information and Knowledge Management (CIKM '06)*, 2006.

[7] J.S. Vitter and M. Wang, "Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '99)*, pp. 193-204, 1999.

[8] R. Agrawal, C. Faloutsos, and A.N. Swami, "Efficient Similarity Search in Sequence Databases," *Proc. Fourth Int'l Conf. Foundations of Data Organization and Algorithms (FODO '93)*, 1993.

[9] M. Garofalakis, J. Gehrke, and R. Rastogi, "Querying and Mining Data Streams: You Only Get One Look. A Tutorial," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '02)*, pp. 635-635, 2002.

[10] J.-H. Lee, D.-H. Kim, and C.-W. Chung, "Multi-Dimensional Selectivity Estimation Using Compressed Histogram Information," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '99)*, pp. 205-214, 1999.

[11] J.S. Vitter, M. Wang, and B. Iyer, "Data Cube Approximation and Histograms via Wavelets," *Proc. Seventh Conf. Information and Knowledge Management (CIKM '98)*, pp. 96-104, 1998.

[12] C.S. Burrus, R.A. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms.* Prentice Hall, 1998.

[13] B. Jawerth and W. Sweldens, "An Overview of Wavelet-Based Multiresolution Analysis," *SIAM Rev.*, vol. 36, no. 3, pp. 377-412, 1994.

[14] A. Haar, "Theorie der Orthogonalen Funktionen-Systeme," *Mathematische Annalen*, vol. 69, pp. 331-371, 1910.

[15] D. Gabor, "Theory of Communication," *J. Inst. Electrical Engineers*, vol. 93, no. 22, p. 429, 1946.

[16] D.L. Donoho, "De-Noising by Soft-Thresholding," *IEEE Trans. Information Theory*, vol. 41, no. 3, pp. 613-627, 1995.

[17] A.C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries," *The VLDB J.*, pp. 79-88, 2001.

[18] G. Colliat, "Olap, Relational, and Multidimensional Database Systems," *SIGMOD Record*, vol. 25, no. 3, pp. 64-69, 1996.

[19] S. Sarawagi, "Indexing OLAP Data," *Data Eng. Bull.*, vol. 20, no. 1, pp. 36-43, 1997.

[20] C.-T. Ho, R. Agrawal, N. Megiddo, and R. Srikant, "Range Queries in OLAP Data Cubes," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '97)*, pp. 73-88, 1997.

[21] J.S. Lim, *Two-Dimensional Signal and Image Processing.* Prentice Hall, 1990.

[22] Y. Matias, J.S. Vitter, and M. Wang, "Dynamic Maintenance of Wavelet-Based Histograms," *The VLDB J.*, pp. 101-110, 2000.

[23] K.-P. Chan and A.W.-C. Fu, "Efficient Time Series Matching by Wavelets," *Proc. 15th IEEE Int'l Conf. Data Eng. (ICDE '99)*, 1999.

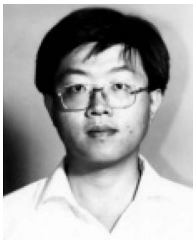
[24] H.E. Hurst, "Long-Term Storage Capacity of Reservoirs," *Trans. Am. Soc. of Civil Engineers*, p. 770, 1951.

[25] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing.* Prentice Hall, 1975.

[26] M. Garofalakis and P.B. Gibbons, "Wavelet Synopses with Error Guarantees," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '02)*, 2002.

[27] TPCD, TPC Benchmark, 1995.

Ming-Jyh Hsieh received the BS degree from the Department of Physics, National Taiwan University, Taipei, and the PhD degree in electrical engineering from the National Taiwan University, Taipei, in 2006. His research interests include data mining, data warehousing, and database.



Ming-Syan Chen received the BS degree in electrical engineering from the National Taiwan University, Taipei, and the MS and PhD degrees in computer, information, and control engineering from the University of Michigan, Ann Arbor, in 1985 and 1988, respectively. He was the chairman of the Graduate Institute of Communication Engineering (GICE), National Taiwan University, from 2003 to 2006. He is now a distinguished professor jointly appointed by the Electrical Engineering Department, Computer Science and Information Engineering Department, and GICE, National Taiwan University. He was a research staff member at IBM T.J. Watson Research Center, New York, from 1988 to 1996. He served as an associate editor of the *IEEE Transactions on Knowledge and Data Engineering* from 1997 to 2001 and is currently on the editorial board of the *Very Large Data Base (VLDB) Journal* and *Knowledge and Information Systems*. His research interests include database systems, data mining, mobile computing systems, and multimedia networking, and he has published more than 230 papers in his research areas. He is a recipient of the National Science Council (NSC) Distinguished Research Award, Pan Wen Yuan Distinguished Research Award, Teco Award, Honorary Medal of Information, and K.-T. Li Research Breakthrough Award for his research work, as well as the IBM Outstanding Innovation Award for his contribution to a major database product. He also received numerous awards for his teaching, inventions, and patent applications. He is a fellow of the ACM and the IEEE.

Philip S. Yu received the BS degree in electrical engineering from the National Taiwan University, Taipei, the MS and PhD degrees in electrical engineering from Stanford University, Stanford, California, and the MBA degree from New York University, New York. He is currently the manager of the Software Tools and Techniques Group, IBM T.J. Watson Research Center. He is an associate editor of the *ACM Transactions on the Internet Technology*, is a member of the IEEE Data Engineering Steering Committee, and is also on the steering committee of the IEEE Conference on Data Mining. He was the editor-in-chief of the *IEEE Transactions on Knowledge and Data Engineering* from 2001 to 2004 and an editor, advisory board member, and guest coeditor of the special issue on mining of databases. He has also served as an associate editor of *Knowledge and Information Systems*. He has been a program chair, cochair, or committee member of many international conferences. His research interests include data mining, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, and performance modeling. He has published more than 400 papers in refereed journals and conference proceedings and is the holder or has applied for more than 250 US patents. He is an IBM Master Inventor and a fellow of the ACM and the IEEE. He has received several IBM honors, including two IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, two Research Division Awards, and the 84th Plateau of Invention Achievement Award. He received an Outstanding Contributions Award in the IEEE International Conference on Data Mining in 2003 and also an IEEE Region 1 Award for promoting and perpetuating numerous new electrical engineering concepts in 1999.