Efficient Process of Top-k Range-Sum Queries over Multiple Streams with Minimized Global Error

Hao-Ping Hung, Kun-Ta Chuang, *Member*, *IEEE*, and Ming-Syan Chen, *Fellow*, *IEEE*

Abstract—Due to the resource limitation in the data stream environments, it has been reported that answering user queries according to the wavelet synopsis of a stream is an essential ability of a Data Stream Management System (DSMS). In the literature, recent research has been elaborated upon minimizing the local error metric of an individual stream. However, many emergent applications such as stock marketing and sensor detection also call for the need of recording multiple streams in a commercial DSMS. As shown in our thorough analysis and experimental studies, minimizing global error in multiple-stream environments leads to good reliability for DSMS to answer the queries. In contrast, only minimizing local error may lead to a significant loss of query accuracy. As such, we first study in this paper the problem of maintaining the wavelet coefficients of multiple streams within collective memory so that the predetermined global error metric is minimized. Moreover, we also examine a promising application in the multistream environment, that is, the queries for top-k range sum. We resolve the problem of efficient top-k query processing with minimized global error by developing a general framework. For the purposes of maintaining the wavelet coefficients and processing top-k queries, several well-designed algorithms are utilized to optimize the performance of each primary component of this general framework. We also evaluate the proposed algorithms empirically on real and simulated data streams and show that our framework can process top-k queries accurately and efficiently.

Index Terms—Data Stream Management System, top-k queries, wavelet synopses.

1 Introduction

Due to the popularity of data-intensive applications, the technology in a Data Stream Management System (DSMS) becomes an emerging research topic [1]. Compared to the conventional data sets, which are persistently stored in the database, the data cells in an unbounded stream may arrive rapidly and unpredictably. To provide online services in various streaming applications such as network monitoring, financial marketing, Web page visits, and sensor readings, we can only get one look at data without storing the whole stream in the memory. To achieve this, several techniques such as wavelets [15], [17], [24], histogram analysis [20], regression model [29], and sampling [11] are applied for sketching the unlimited data streams within limited memory space.

Among all of the sketching techniques, the wavelet-based approaches attract the most research attention due to their excellent energy compaction and decorrelation properties. Wavelet transform (also known as wavelet decomposition) provides effective data reduction in streaming environments. In general, the wavelet synopsis of a stream is extracted by applying the wavelet transform on an input collection and then summarizing the stream by retaining only a selected subset of the produced wavelet coefficients. The original data can thus be approximately reconstructed based on this compact synopsis. Previous research [27], [32] has shown that storing most representative wavelet coefficients can achieve accurate approximation. In research fields relating to the wavelet synopses, given different error metrics such as the L^2-*norm average error* [15], [18], [27], the *maximum absolute error* [24], the *maximum relative error* [24], or even the *weighted* L^p-*norm error* [17], the DSMS can provide accurate answers to these *value-based* queries[1] according to the synopses (for example, "Select the average stock price of the IBM Corp. last month" or "Select the temperature of the LA downtown on 1 January 2006").

However, the DSMS designed for conventional value-based queries cannot retrieve the relativity among various streams. Note that under many circumstances, the rank of the streams will be more meaningful to users than the value of each stream. The queries sent by these users who expect to receive a set of entities are referred to as *entity-based* queries [9], [10]. One common entity-based query is the top-k query, which aims at finding out the k entities with the largest summation of all attributes within a specific range.[2] For example, a user may issue a query like "Select the router that suffers most from traffic congestion last minute" or "Select the top-five sensors that detect the highest average temperature yesterday." The impact of

- *H.-P. Hung and K.-T. Chuang are with the Graduate Institute of Communication Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei, Taiwan, ROC. E-mail: {hphung, doug}@arbor.ee.ntu.edu.tw.*
- *M.-S. Chen is with the Department of Electrical Engineering and the Graduate Institute of Communication Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei, Taiwan, ROC. E-mail: mschen@cc.ee.ntu.edu.tw.*

Manuscript received 4 June 2006; revised 27 Dec. 2006; accepted 15 Mar. 2007; published online 23 May 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0288-0606.
Digital Object Identifier no. 10.1109/TKDE.2007.1065.

1. The value-based query is regarded as the query for the numerical value (for example, mean and variance) of a specific stream [27], [32].
2. In this paper, we use the range-sum operator due to its popularity in many applications compared to the other classes of range queries.

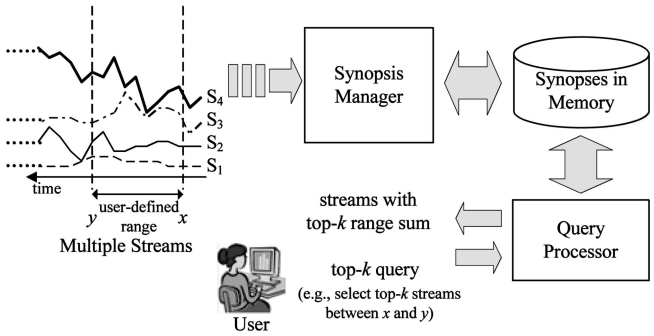


Fig. 1. Model of the top-k query processing system.

such top-k queries will be even noticeable when the DSMS concurrently monitors a massive number of streams. Motivated by the great importance of entity-based queries under the circumstances of multiple streams, we address in this paper the problem of processing the top-k queries according to the retained synopses of these streams.

1.1 Problem Description

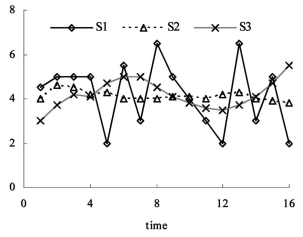
The architecture of our top-k query processing system is shown in Fig. 1. There are three primary components: 1) the synopsis manager, 2) the collection of the wavelet synopses, and 3) the query processor. With the multiple streams as the input, the query processing system will answer the top-k queries within a specific range. The synopsis manager is responsible for calculating the wavelet coefficients as the streams evolve over time. The wavelet synopses of multiple streams share the collective space in the main memory. When the number of stored coefficients exceeds the predetermined threshold B, the synopsis manager will discard the redundant coefficients according to the eviction policies for the different error metrics. In addition, the query processor will return the top-k entities, that is, the k streams with the largest range sums.

In this paper, we first explore the problem of minimizing the global error when monitoring multiple streams. Global optimization is proved to be an important issue in optimizing the overall quality in many applications [7], [23]. In the streaming environment, we will further show that minimizing the global error will enhance the accuracy of processing top-k queries. The importance of minimizing the global error can be best understood by the following example. Suppose that we monitor three data streams $\{S_q, 1 \leq q \leq 3\}$ from time points t_1 to t_{16}. The original data cells of each stream are listed in Fig. 2a. Fig. 2b shows the trend of each stream. Consider that at most 15 wavelet coefficients can be retained in the memory. To retain the wavelet coefficients, we first implement a direct extension from conventional approaches, which minimize the error of an individual stream (that is, local error). In this approach, the 15 quotas are evenly distributed to each stream (that is, $B(S_1) = B(S_2) = B(S_3) = 5$), and the conventional local error minimization approaches are adopted. Fig. 2c shows the reconstruction from the synopses. Without loss of generality, we employ the algorithm proposed in [31], which minimizes the L^2-norm average error. On the other hand, if we assign different quotas to different streams, for example, $B(S_1) = 10$, $B(S_2) = 1$, and $B(S_3) = 4$, due to the variety of evolving speed, the reconstructed streams are shown in Fig. 2d. By comparing Figs. 2c and 2d, we observe that a significant amount of information is lost in Fig. 2c. Moreover, during the intervals $[t_3, t_4]$ and $[t_9, t_{12}]$, the rank of the reconstructed streams is different from that in the original streams. This phenomenon will degrade the quality of answering the entity-based queries (for example, the top-1 stream between $[t_9, t_{12}]$ is S_1 instead of S_2). Therefore, it is necessary for the synopsis manager to design an approach to minimize the error globally.

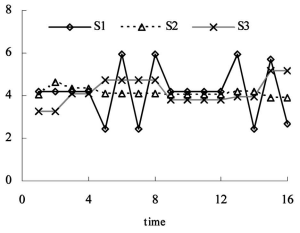
time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
S_1	4.5	5	5	5	2	5.5	3	6.5
S_2	4	4.6	4.5	4.2	4.3	4	4	4
S_3	3	3.7	4.2	4.1	4.7	5	5	4.5

time	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}
S_1	5	4	3	2	6.5	3	5	2
S_2	4.1	4.1	4	4.2	4.3	4	3.9	3.8
S_3	4.1	3.8	3.6	3.5	3.7	4.1	4.7	5.5

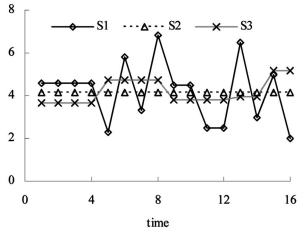
(a)



(b)



(c)



(d)

Fig. 2. The issue of minimizing the global error. (a) Original data cells of the streams. (b) Original data streams. (c) Reconstructed streams with $B(S_1) = 5$, $B(S_2) = 5$, and $B(S_3) = 5$. (d) Reconstructed streams with $B(S_1) = 10$, $B(S_2) = 1$, and $B(S_3) = 4$.

Another important issue is to efficiently process the top-k queries from the wavelet synopses. Since the data streams evolve rapidly, a query processor should provide online services to DSMS users. In this paper, we aim at calculating the top-k range sum in such a way that the cost of accessing the memory can be minimized without trading accuracy of the query answers. To efficiently process the top-k queries, the objective of this paper can be specified as follows: Given the wavelet SYM and a specific range defined by the user, find out the k streams with the largest range sum accurately in such a way that the number of accessed coefficients can be minimized.

1.2 Challenges

The issue of processing top-k queries over multiple streams with minimized global error is important and challenging. In general, the following challenges are inevitability encountered:

1. In the synopsis manager, how should we calculate the wavelet coefficients as multiple streams evolve? It is noted that in a centralized DSMS, a general and flexible way of distributing the main memory is to let the synopses of all streams share the collective memory. That is, there are only a fixed number of

total wavelet coefficients for all streams. Since the trends of the streams vary over time, it is difficult to distribute the memory space adaptively so that the global error can be minimized. A direct solution is to apply the conventional approaches [15], [17], [18], [24] for local error metrics and distribute the total coefficient evenly to each stream. However, as we mentioned above, such a static method will incur an even larger global error and will tend to result in quality loss of query results.

2. How should we arrange the retained wavelet coefficients in the memory? A well-designed data structure is helpful for efficient insert and discard operations on wavelet coefficients. Moreover, the efficiency of processing top-k queries will depend on the design of the data structure.
3. How should the system respond when the user gives his/her range of interest? From [32], we know that only part of the coefficients will contribute to the range sum for each individual stream. However, how we can apply this property to the data structure storing the coefficients of multiple streams remains challenging.
4. Also, note that high memory access cost will degrade the performance of processing online top-k queries. It is still challenging to solve the problem of how we can determine the order of accessing the coefficients and obtain the accurate answer to the top-k query with minimized memory access.

1.3 Framework Description

To resolve the difficulties mentioned above, we propose in this paper a general framework named Top-k Over Multiple Streams (TOMS) to process top-k queries over multiple streams. Based on the architecture in Fig. 1, several well-designed algorithms are employed to optimize the performance of each component of the query processing system.

For challenge 1, we propose the SYM procedure to maintain multiple streams. With multiple streams as the input, the SYM procedure first builds the wavelet tree dynamically as the streams evolve. If the total number of wavelet coefficients among all streams exceeds a predetermined bound B, then the SYM procedure will discard the redundant coefficients that lead to the minimum error propagation with respect to a given global error metric.

To retain the wavelet coefficients, we design a novel data structure $W\text{-}Lists$. In $W\text{-}Lists$, the wavelet coefficients are classified into several categories. Each category is associated with a weighting factor to reflect the user-defined range. We also build two types of indices in $W\text{-}Lists$ so as to facilitate the memory access. Note that $W\text{-}Lists$ is able to overcome challenge 2 and challenge 3.

To minimize the memory access when a top-k query is processed, we propose the PArallel search With Adaptivity (PAWA) algorithm to access the coefficients stored in the $W\text{-}Lists$. The PAWA algorithm has the following important characteristics. First, the PAWA algorithm prunes the searching space by neglecting the coefficients that do not contribute to the range sum. Second, the lower bound of the top-k range sum will be evaluated when PAWA is executed. This property enables us to obtain the top-k streams without accessing all the coefficients of $W\text{-}Lists$. Moreover, PAWA utilizes an adaptive searching strategy, which will lead to better chances of finding exact top-k streams. Compared to other competitive approaches, the PAWA algorithm achieves the minimum cost of memory access. Note that PAWA is able to solve challenge 4 mentioned in the left column.

1.4 Our Contributions

Overall, in this paper, our contributions are threefold:

1. We propose an efficient procedure to obtain the wavelet synopses when multiple streams are monitored. Different from the approaches extended from prior research, which only focus on minimizing the local error, our monitoring scheme can achieve minimized global error, which will significantly enhance the accuracy of processing the top-k queries over multiple streams.
2. When the wavelet coefficients are retained, a casual arrangement may cause inefficiency in processing top-k queries. To store the coefficients in wavelet synopses, we propose the W-Lists structure for efficient insertion/deletion and processing of top-k queries.
3. Motivated by the observation that the users tend to be interested in the top-ranked results in the multi-stream environments, we explore the problem of processing top-k queries, for example, "Select the top five companies with the highest average stock prices from 10:00 a.m. to 11:00 a.m. on 20 December 2006." To achieve high efficiency in accessing the wavelet coefficients in W-Lists, we propose the PAWA algorithm. Compared to the naive approach, which may access all coefficients in W-Lists, the PAWA algorithm needs to access only a small part of coefficients before the correct answers are obtained.

1.5 Outline

The rest of this paper is outlined as follows: In Section 2, preliminaries, including the wavelet basics, the related works, and an overview of the TOMS framework, are given. In Section 3, we design the SYM procedure and $W\text{-}Lists$ to maintain the wavelet coefficients. In Section 4, we describe the details of the PAWA algorithm and the other competitive approaches. The experimental results are shown in Section 5 and, finally, this paper concludes with Section 6.

2 Preliminaries

2.1 Review of Wavelet Basics

Theoretically, the wavelet transform represents a function in terms of a coarse overall approximation and a series of detail coefficients that revise the function from coarse to fine. Among all wavelet transform techniques, the Haar wavelet basis representation is the most usual way in decomposing a sequence of data values.[3] The concept of Haar wavelet transform can be best understood by the following example.

3. The theorem of Haar wavelets is out of the scope of this paper. Interested readers are referred to [31].

TABLE 1
Wavelet-Based Multiresolution Analysis

Resolution	Averages	Detail Coefficients
3	$\{4, 2, 6, 4, 9, 6, 5, 1\}$	$-$
2	$\{3, 5, 7.5, 3\}$	$\{1, 1, 1.5, 2\}$
1	$\{4, 5.25\}$	$\{-1, 2.25\}$
0	$\{4.625\}$	$\{-0.625\}$

Example 1. Suppose we have the numerical time series $A = \{4, 2, 6, 4, 9, 6, 5, 1\}$. For multiresolution analysis, we first average the values pairwise to get a low-resolution signal. Therefore, we have the average values {3, 5, 7.5, 3}, where the first two values in A, that is, 4 and 2, are averaged to 3, and so on. To avoid losing any information in the averaging process, the difference values {1, 1, 1.5, 2} are also stored. Recursively applying the above pairwise averaging and differencing process on the lower resolution array containing the averages, we get the full decomposition, as shown in Table 1. The wavelet transform of A is the single coefficient representing the overall average of the data values followed by the detail coefficients in the order of increasing resolution. Thus, the one-dimensional Haar wavelet transform of A is given by $W_A = \{4.625, -0.625, -1, 2.25, 1, 1, 1.5, 2\}$.

The error tree structure, proposed by Matias et al. [27], is a helpful tool for exploring and understanding the key properties of the Haar wavelet decomposition. Fig. 3 depicts the illustration of an error tree structure. Each internal node c_i $(i = 1, \ldots, 16)$ is associated with a wavelet coefficient value, and each leaf d_i $(i = 1, \ldots, 16)$ is associated with a value in the original data cells. Taking A and W_A mentioned above as examples, c_1 corresponds to the first value in W_A, that is, 4.625, and so on. Likewise, d_1 represents the first value in A, that is, 4, and so on. Note that the wavelet coefficients carry different weights with respect to their importance in rebuilding the original data values. In order to equalize the importance of all wavelet coefficients, we need to normalize the wavelet coefficients appropriately. A common normalization scheme is to divide each coefficient by $\sqrt{2^l}$, where l denotes the corresponding level in the error tree to the coefficient. For example, let W_A^* denote the set of normalized coefficients in W_A. We can obtain $W_A^* = \{4.625, -0.625, -0.707, 1.591, 0.5, 0.5, 0.75, 1\}$.

Given an error tree T and an internal node c_k of T, $k > 1$, several special terms are defined as follows:

Definition 1. *The set of left leaves (respectively, right leaves), denoted by $lleaves$ (respectively, $rleaves$) is defined as the set of leaf nodes in the subtree rooted at c_k's left (respectively, right) child.*

Definition 2. *The set of path nodes of c_k (respectively, d_k), denoted by $path_k$, is defined as the set of all nonzero internal nodes in T, which lie on the path from c_k (respectively, d_k) to the root of T.*

Definition 3. *The range sum between d_l and d_h, denoted by $d(l:h)$, is defined as $\sum_{i=l}^{h} d_i$.*

According to [32], we can calculate the range sum between d_l and d_k directly from the wavelet coefficients without reconstructing each leaf d_i $(l \le i \le h)$. More specifically, $d(l:h) = \sum_{c_j \in path_l \cup path_h} x_j$, where

$$x_j = \begin{cases} (h - l + 1) * c_j, & \text{if } j = 1 \\ (|(LL(j,l,h)| - |RL(j,l,h)|) * c_j, & \text{otherwise,} \end{cases} \quad (1)$$

where $LL(j,l,h) = lleaves_j \cap \{d_l, d_{l+1}, \ldots, d_h\}$ and $RL(j,l,h)$ is defined similarly.

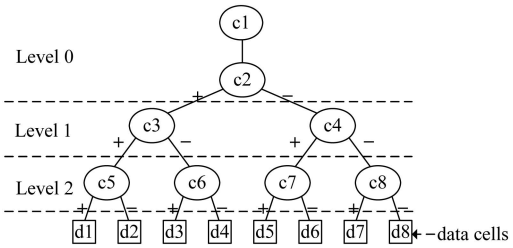


Fig. 3. An error tree structure for wavelet synopsis.

Let $\widehat{d}_i$ (respectively, d_i) denote the reconstructed (respectively, original) data value of cell i. Several common error metrics, which are popularly utilized in prior works, are given as follows:

Definition 4. *The L^2-norm* average error (L^2_err), *the* maximum absolute error (max_abs), *and the* maximum relative error (max_rel) *are defined as* $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - \widehat{d}_i)^2}$, $\max_{i=1}^{N}\{|d_i - \widehat{d}_i|\}$, *and* $\max_{i=1}^{N}\{\frac{|d_i - \widehat{d}_i|}{\max\{|d_i|, S\}}\}$, *respectively. Note that N represents the number of cells, and S is referred to as the* sanity bound *[14], which is used to prevent the influence of very small values in the aggregate error.*

2.2 Related Works

Recent works have demonstrated the effectiveness of wavelet decomposition in reducing large amounts of data to compact sets of wavelet coefficients. Given the target number B, the wavelet thresholding problem is the problem of selecting B wavelet coefficients in such a way that a predetermined error metric can be minimized. In [31], Stollnitz et al. show that retaining the B largest absolute coefficients (normalized) will achieve the optimal solution in minimizing the L^2-*norm average error*. As for the maximum-error metrics such as the *maximum absolute error* and the *maximum relative error*, Garofalakis and Kumar show in [14] that this kind of problem can be solved optimally via the concept of *dynamic programming*. To approximate the optimal solution in maximum-error metrics, Karras and Mamoulis propose two algorithms GreedyAbs and GreedyRel in [24] to derive the near-optimal wavelet synopses with much lower computing complexity. In [13], Garofalakis and Gibbons introduce the probabilistic wavelet synopses, which provide *unbiased* answers in minimizing the different error measures, including the L^2-*norm average error*, the *maximum absolute error*, and the *maximum relative error*.

Different from the conventional wavelet thresholding algorithms, which may be executed offline, the one-pass-scan property restricts the thresholding algorithms for streaming data to be executed in an online mode. To generate the wavelet synopses from streaming data, Gilbert et al. [15] study the wavelet synopses of aggregate values with minimized L^2-*norm average error*, and the authors in

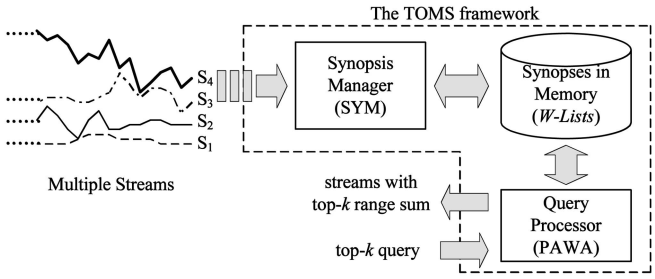


Fig. 4. System architecture of the TOMS framework.

[18] extend this approach for data with multiple measures. In [24], Guha et al. also extend the concept of their greedy algorithms to the streaming environment in minimizing the maximum-error metrics. In [17], Guha and Harb further construct the wavelet synopses for data streams in minimizing the noneuclidean error metrics, including the weighted L^p and relative L^p error metrics. As for multidimensional synopses, Hsieh et al. [22] integrate the Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) in order to provide more accurate approximation than the conventional DWT. It is noted that the streaming data can also be monitored within a sliding window. Several approaches relating to such window-based wavelet synopses are found in [5] and [6].

Top-k query processing has received a great deal of attention in various research fields such as similarity search on multimedia [8], middleware [12], peer-to-peer (P2P) networks [3], distributed query [28], Web technologies [4], and so forth. Several top-k problems are also addressed in stream monitoring [2], [25]. Babcock and Olston [2] aims at processing user queries streams and finding out the top-k visited sites in a distributed hit-sources environment. In [25], Koudas et al. investigate the k nearest neighboring streams, given a predetermined error bound e. It is noted that in [2], [25], the authors aim at monitoring the behavior of the data streams instead of processing the user query. Therefore, the problem addressed in this paper is intrinsically different from those in [2], [25].

2.3 System Architecture

In this paper, we optimize each component in Fig. 1, and the relevant methods are shown in Fig. 4. With the multiple streams as the input, the TOMS framework will answer the top-k query within a specific range given by the user. For brevity, in this paper, we only discuss the global optimization and top-k query processing problems in the wavelet representation domain. However, the concepts of global optimization and top-k query processing in this paper can be applied on the histogram construction [19], [20], which is also a well-known technique in monitoring data streams.[4]

Several notations used throughout this paper are listed in Table 2. Consider M streams evolving as time goes on. A

4. It is noted that the wavelet-based synopses and the histogram-based synopses have several characteristics in common, since for each wavelet-based representation, we can transform it into a specific histogram-based representation. Therefore, our concepts in the wavelet domain can be applied to the histogram domain. However, for other approaches of generating synopses, our approaches may incur compatibility problems. For example, consider the Singular Value Decomposition (SVD)-based representation in [16]. The data cells of multiple streams are viewed as an $M \times N$ matrix. Moreover, the synopses are used to summarize the correlation between two data streams instead of the trend of one data stream. Therefore, our approaches cannot be applied to this scenario.

TABLE 2
Description of the Symbols

Symbols	Semantics
N	Number of the data cells in a stream
M	Number of the streams
B	Target number of total coefficients retained in memory
C_B	The coefficients retained in memory
$d_{i,j}$	Data value of j-th cell in i-th stream
$\widehat{d}_{i,j}$	Reconstructed (approximate) value $d_{i,j}$
$n(s,l,p)$	The coefficient at the p-th placement of level l in the forest corresponding to stream s
$c(s,l,p)$	The value of $n(s,l,p)$
L^2_err	The L^2-norm average error
max_abs	The maximum absolute error
max_rel	The maximum relative error

predetermined integer B indicates the target number of total coefficients retained in memory. These coefficients form a collection C_B. Using the Inverse Discrete Wavelet Transform (IDWT), we reconstruct the data value of the jth cell in the ith stream (that is, $d_{i,j}$) as $\widehat{d}_{i,j}$. The error metrics in Definition 4 can be generalized to model the global error in a multistream environment, as shown in Definition 5.

Definition 5. *Considering the multistream environment, let $\widehat{d}_{i,j}$ (respectively, $d_{i,j}$) denote the reconstructed (respectively, original) data value of cell j in stream i. Extended from Definition 4, the* L^2-norm average error (L^2_err), *the* maximum absolute error (max_abs), *and the* maximum relative error (max_rel) *are defined as*

$$L^2\_err = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(d_{i,j}-\widehat{d}_{i,j}\right)^2},$$

$$max\_abs = \max_{i=1}^{M}\max_{j=1}^{N}\left\{|d_{i,j}-\widehat{d}_{i,j}|\right\},$$

$$max\_rel = \max_{i=1}^{M}\max_{j=1}^{N}\left\{\frac{|d_{i,j}-\widehat{d}_{i,j}|}{\max\{|d_{i,j}|,S\}}\right\},$$

where S and N are defined similarly to those in Definition 4.

Note that different error metrics are suitable for different applications. For the purpose of suiting the TOMS framework to different applications, our objective in this paper is to minimize the global error, given the predetermined error metrics in Definition 5.

3 Monitoring Multiple Streams

3.1 Synopses of Multiple Streams

There are several concerns as we seek to maintain the wavelet coefficients over multiple streams within the limited memory. First, in the streaming environments, the conventional error tree structure [27], which identifies the lowest level of an error tree with the maximum level number, is irrational. Moreover, since the synopsis manager should response immediately as new data cells come, the conventional error tree architecture only capable of representing 2^n data cells is not suitable here. Therefore, to maintain the wavelet coefficients, we represent the

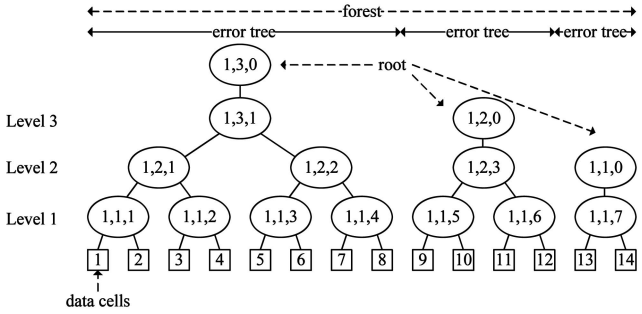


Fig. 5. The forest representation of each stream.

wavelet coefficients of each stream as a *forest*, which corresponds to a set of error trees. We identify each node in a specific forest by using three attributes: the identifier of stream, the level number, and the *placement*. Specifically, each node denoted by $n(s,l,p)$ represents the wavelet coefficient at the pth placement of level l in the forest of stream s. An illustrative example of the forest representation is shown in Fig. 5.

The identification of the wavelet coefficients is based on the following rules:

1. The placement number of the root of each error tree in a forest is set to be 0.
2. Since the height of each error tree will increase as the stream evolves, we use the bottom-up method to represent the level. For the coefficient that is generated from the data cells, the level number is set to be 1. As for the coefficient that is generated from the roots in level l, the level number is set to be $l+1$.
3. The *placement* of all nodes except the roots is used to record the number of the nodes that appear in a specific level of a forest. Consider that there are p nodes in level l. The *placement* number of the next node appearing in level l is set to be $p+1$.

A forest is constructed according to the following principles. First, the error trees in a forest are of different heights. Moreover, given a specific data stream, when each data cell is received, this data cell will be first kept in the buffer. Once there are exactly two data cells in the buffer, an error tree with one average node (that is, root node) and one error node (that is, internal node) is constructed. According to the definition in Section 2.1, the value of the average node (respectively, the error node) is $\frac{u+v}{2}$ (respectively, $\frac{u-v}{2}$), where u and v are the values of two data cells in the buffer. Since the data cells are received continuously, more error

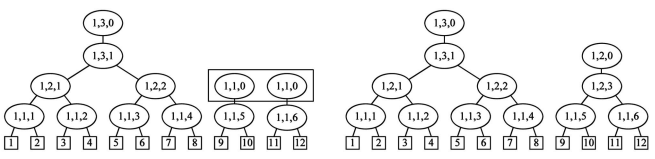


Fig. 6. The merging operation in the forest structure. (a) Before the merging operation. (b) After the merging operation.

Procedure SYM
input: B, $d_1 \sim d_M$ (the incoming data cells of stream $1 \sim M$),
C_B (the coefficients retained in memory)
output: C_B' (the coefficients retained in memory after update)
1 Insert the data cells into a temporary buffer
2 **for** $i = 1$ to M
3 **if** the temporary buffer contains exactly two data cells of stream i
4 Build an error tree using the two cells
5 **while** there are two roots with the same level
6 Remove the roots from C_B
7 Merge the two corresponding trees of the roots
8 Insert the new coefficients into C_B
9 **while** the total number of coefficients is larger than B
10 Discard the coefficient leading to minimum error propagation from C_B
11 **Return** the updated C_B

Fig. 7. Algorithmic form of the SYM procedure.

trees will be constructed. Once we find that there are two error trees with the same height, as shown in Fig. 6a, the roots will be merged to a new average node and a new error node. In Fig. 6b, after the merging operation, a new error tree is constructed. Let U and V denote the values of two original roots. The value of the new average node $n(1,2,0)$ will be $\frac{U+V}{2}$, whereas the value of the new error node $n(1,2,3)$ will be $\frac{U-V}{2}$.

Using this representation, we give each coefficient a distinct identifier. As the number of wavelet coefficients grows as streams evolve, we shall retain the most representative coefficients within limited memory. It is noted that the memory space is shared by all streams. In order to best distribute (assign) coefficients to different streams, we design the SYM procedure, which is outlined in Fig. 7.

The SYM procedure will be triggered right after the synopsis manager receives data cells.[5] Let d_i represent the incoming data cell of the stream i. The SYM procedure will first build the wavelet tree corresponding to each stream i by using d_i. If there are two roots with the same level, then the wavelet tree will be built recursively until all roots belong to different levels. To achieve the normalization of the wavelet bases while the stream advances, we use $(u+v)/\sqrt{2}$ and $(u-v)/\sqrt{2}$ to calculate the *average node* and the *difference node* of a wavelet tree, given the values u and v of the two roots to be merged. This concept is the same as that of algorithm A_l in [18]. Such representations are different from $(u+v)/2$ and $(u+v)/2$ for the conventional wavelet tree. Although it will make each coefficient overmagnified with the factor $\sqrt{2^{MaxHeight}}$, where $MaxHeight$ denotes the maximum height among all wavelet trees in the forest, this effect will be eliminated when the top-k queries are processed, which will be discussed in Section 3.2.

When the number of retained coefficients exceeds the predetermined threshold B, SYM starts discarding redundant coefficients. In order to effectively distribute B coefficients into multiple streams, the thresholding policy is based on the global error propagation among all streams.

5. Note that we describe our model in the case that data cells of different streams are received simultaneously. In practice, the same concept can be easily extended to fit the case that different data cells come at different time points. It is an implementation matter and is thus skipped here for ease of exposition.

Considering a specific error tree, once a wavelet coefficient $n(s,l,p)$ is discarded, the data cells corresponding to the leaves of $n(s,l,p)$ will incur reconstruction errors in global error metrics in Definition 5. The error propagation of $n(s,l,p)$ can be viewed as the aggregation of the reconstruction error for each data cell in the leaf. For example, given the L^2-norm average error as the metric, a normalized coefficient with minimum absolute value leads to minimum error propagation. As for the maximum absolute error and the maximum relative error, we can use the algorithms proposed in [24] to approximate the error propagation of each coefficient and discard the node with minimum error propagation. The wavelet coefficient with the minimum error propagation will be discarded until the memory limit is satisfied. Using this policy, since the data streams with less variation will have smaller error propagation, they will be assigned smaller numbers of wavelet coefficients.[6]

The SYM procedure is based on the *landmark* model [30]. In the landmark model, the monitored data streams are with fixed starting time as the landmark. On the other hand, the end time will be increased as the streams evolve. For some special purposes, SYM can also be modified to suit the *sliding-window* model, in which only the recent data cells are monitored. To generate the wavelet synopses with minimized global error in a sliding window, we combine the SYM procedure and DCIF algorithm [26]. The integrated procedure for sliding window is outlined as follows. When new data cells are received, we will use the delete operations in DCIF to discard the nodes that are out of the sliding window for each stream. After that, the new error trees will be constructed using the SYM procedure. Since the number of newly generated nodes may exceed that of discarded nodes, we will employ the SYM procedure to discard the nodes with minimum error propagation until B coefficients are retained.

Complexity analysis. The space complexity of the SYM procedure is $O(B)$. The time complexity differs, given different error metrics. When we use the *L^2-norm average error*, the time complexity is $O(\log B)$. However, given the *maximum absolute error* or the *maximum relative error*, the time complexity will be $O(M \log^2 N)$. Specifically, for the maximum absolute error and the maximum relative error, we can implement a heap structure on the wavelet coefficients. The coefficients in the heap structure are sorted according to the amount of error propagation. Specifically, it costs $O(\log^2 N)$ to calculate the error propagation for each data stream and $O(\log B)$ for heap insertion. Since $O(\log^2 N)$ may dominate the complexity when N is much larger than B, we neglect the effect of $O(\log B)$.

3.2 Example of SYM

To best understand our model, we give an example in this section to describe the execution of the TOMS framework.

6. The best discarding algorithm is to store the whole data cells by using an infinite buffer, generate the complete error tree for each data stream, and discard the wavelet coefficient with global minimum error propagation until the memory limit is satisfied. However, in the streaming environments, this offline method is not practical. Therefore, based on the same intuition, we develop the SYM procedure to perform online discarding in the streaming environments. In our experimental results, we will show that our SYM procedure achieves very similar quality to the offline method.

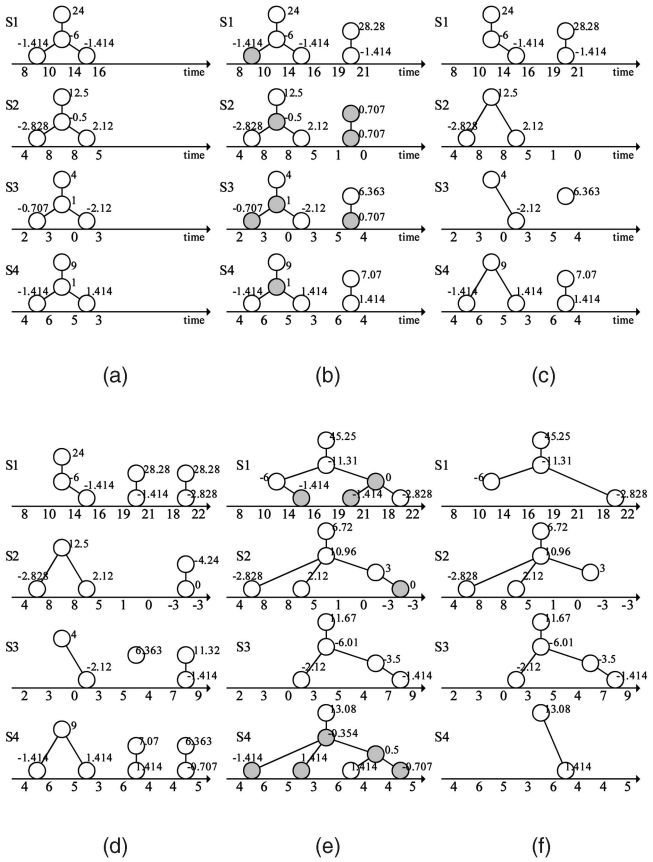


Fig. 8. Execution scenario of the SYM procedure. (a) After four values. (b) After six values. (c) Removing redundant coefficients. (d) After eight values. (e) Merging two subtrees with the same height. (f) Removing redundant coefficients.

Let t_i denote the time stamp and S_j identify the stream. Consider four data streams $\{S_1, S_2, S_3, S_4\}$, which evolve from t_1 to t_8. Suppose that only 16 wavelet coefficients can be stored in memory, that is, $B = 16$. Fig. 8 shows how the coefficients are retained and discarded by the SYM procedure. For ease of exposition, we assume that each stream comes with the same rate. Initially, SYM retains all the coefficients with its best effort. SYM keeps exactly 16 coefficients at t_4 (four for each stream), as depicted in Fig. 8a. At t_6, since the new wavelet tree of each stream causes the number of retained coefficients exceeding B, some of the coefficients should be discarded. The SYM procedure will thus discard the nodes with minimum error propagation. Note that in this example, we use the *L^2-norm average error* as the error metric. The shadow in Fig. 8 represents the nodes that will be discarded. Fig. 8c depicts the error trees after these nodes are evicted. At t_8, as new data cells come, we find that the forest of each stream contains two trees with the same height, as shown in Fig. 8d. Under this circumstance, the SYM procedure will first merge the trees with the same height recursively until all roots belong to different levels. After that, the redundant coefficients will be evicted. Figs. 8e and 8f show the wavelet trees after merging the root and after discarding the coefficients, respectively.

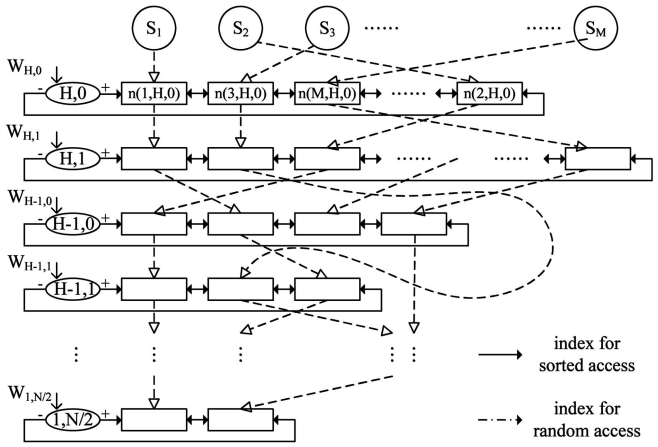


Fig. 9. Structure of the $W\text{-}Lists$.

3.3 The W-Lists Structure

Although the B coefficients selected by the SYM procedure can be arranged arbitrarily in the memory, a casual arrangement may cause all coefficients to be accessed before the answer to the top-k query is calculated, which will degrade the efficiency of the query processor in performing online top-k queries. Moreover, the data structure should reflect the range specified by the user. To resolve the problems mentioned above, we propose the data structure $W\text{-}Lists$ to retain the wavelet coefficients. The insights of $W\text{-}Lists$ are explained as follows: First, we note that for each forest corresponding to a stream, only the coefficients lying at the specific positions contribute to the range sum. Therefore, we categorize all of the retained coefficients according to their positions in the forest. Moreover, motivated by the observation that the range sum of a stream is regarded as the weighted aggregation of the wavelet coefficients, we associate each category with a weighting factor. To access the coefficients more efficiently, we further design the index structure for *sorted access* and *random access*. The structure of $W\text{-}Lists$ is illustrated in Fig. 9.

The major components of the $W\text{-}Lists$ are described as follows:

1. **Category**. Each category is identified by two attributes: level number l and placement number p. We put the coefficient $n(s,l,p)$ into the category (l,p) for each stream s.
2. **Coefficient**. The coefficients in a specific category are sorted in a descending order.
3. **Index**. In the $W\text{-}Lists$, we build two types of indices. The first type (represented by solid lines in Fig. 9) enables us to access the values sequentially according to the order in which the coefficients are arranged (that is, the *sorted access*). The second type (represented by dashed lines in Fig. 9) facilitates the access to the coefficients of a specific stream in different categories (that is, the *random access*).
4. **Weight**. The coefficients in a certain category will be multiplied by a weighting factor when they are accessed. The weighting factor will be calculated when the user gives his/her range of interest. Consider the category (i,j) with weighting factor $W_{i,j}$. If $W_{i,j} > 0$ (respectively, $W_{i,j} < 0$), the *sorted access* will be performed in the *forward* (respectively, *backward*) direction, that is, the direction identified by "+" and "−" in Fig. 9, respectively. If $W_{i,j} = 0$, neither *sorted access* nor *random access* will be performed on all coefficients in category (i,j). For example, consider category $(H,0)$ in Fig. 9. If $W_{H,0} > 0$, the first coefficient read by the sorted access will be $n(1,H,0)$. If $W_{H,0} < 0$, the first coefficient read by the sorted access will be $n(2,H,0)$.

$W\text{-}Lists$ is updated dynamically after coefficients are inserted or deleted by the SYM procedure. Although there is some overhead in building the indices of $W\text{-}Lists$, it is worthwhile to save the memory access cost in processing top-k queries by trading this overhead. The weighting factors will be initialized when the user specifies the range of interest. Consider the range between cell x and cell y. The weighting factor $W_{i,j}$ of category (i,j) can be formulated as

$$W_{i,j} = \begin{cases} \frac{(y-x+1)}{\sqrt{2^{MaxHeight}}}, & \text{if } j = 0 \\ \frac{(|LL(i,j,x,y)|-|RL(i,j,x,y)|)}{\sqrt{2^{MaxHeight}}}, & \text{otherwise,} \end{cases} \quad (2)$$

where $|LL(i,j,x,y)|$ represents the intersection of the left leaves of the category (i,j) with the specified range. $|RL(i,j,x,y)|$ is defined similarly. $\sqrt{2^{MaxHeight}}$ is used to compensate the overmagnification of each coefficient in the SYM procedure.

Lemma 1. *Given the range (x,y), the range sum of stream s, denoted by $Sum(s,x,y)$, can be formulated as*

$$Sum(s,x,y) = \sum_{\text{category}(i,j) \in path_x \cup path_y} W_{i,j} c(s,i,j), \quad (3)$$

where $c(s,i,j)$ is the value of $n(s,i,j)$.

Proof. According to Definition 3, we can rewrite the range sum as $Sum(s,x,y) = \sum_{(i,j) \in path_x \cup path_y} x(s,i,j)$, where

$$x(s,i,j) = \begin{cases} \frac{(y-x+1)*c(s,i,j)}{\sqrt{2^{MaxHeight}}} & \text{if } j = 0 \\ \frac{(|LL(i,j,x,y)|-|RL(i,j,x,y)|)*c(s,i,j)}{\sqrt{2^{MaxHeight}}} & \text{otherwise.} \end{cases}$$

Using the definition of $W_{i,j}$ in (2), the range sum can be formulated as

$$Sum(s,x,y) = \sum_{(i,j) \in path_x \cup path_y} W_{i,j} c(s,i,j).$$

□

4 Processing Top-*k* Queries

Given the wavelet coefficients arranged in $W\text{-}Lists$, the goal of the query processor is to select k streams with the largest range sum. It is noted that weighting factors reflect the effect of the range defined by users. Moreover, as presented in (3), the range sum of each stream can be represented by a weighted sum of wavelet coefficients. We can calculate the range sum of each stream by examining the coefficients in different categories. A naive method is applying IDWT and then calculating the range sum of each stream from the reconstructed data cells. However, this approach is computationally prohibitive due to the unlimited stream length.

Instead, we obtain the top-k streams directly from the retained coefficients. To achieve this, we present in this section three alternatives, named as BASIC, PSearch, and PAWA, respectively. Furthermore, by a thorough analytical study, we comment that PAWA can lead to the minimum cost of accessing the memory. We then individually present these approaches.

4.1 BASIC Algorithm

To avoid accessing all coefficients, it is intuitive to retrieve top-k streams by accessing coefficients that exactly contribute to the range sum. Lemma 2 shows the determination of these coefficients.

Lemma 2. *For the category (i,j) that does not contribute to the range sum, we have $W_{i,j}=0$.*

Proof. The proof is given as follows:

1. From Lemma 1, we know that if the category (i,j) does not contribute to the range sum, $(i,j) \notin path_x \cup path_y$.
2. To calculate $W_{i,j}$ for each (i,j) that does not belong to $path_x \cup path_y$, we only need to consider the formulation $W_{i,j}=\frac{(|LL(i,j,x,y)|-|RL(i,j,x,y)|)}{\sqrt{2^{MaxHeight}}}$ in (2), since the formulation $W_{i,j}=\frac{(y-x+1)}{\sqrt{2^{MaxHeight}}}$ is only suitable for the root of an error tree. Note that the root of an error tree must belong to $path_x \cup path_y$.
3. If $(i,j) \notin path_x \cup path_y$, there are only two possibilities:
   - There is no overlap between the range (x,y) and the leaves (either left or right) of (i,j), i.e, $|LL(i,j,x,y)|=0$, and $|RL(i,j,x,y)|=0$.
   - All of the leaves (either left or right) are covered by the range (x,y), which implies that $|LL(i,j,x,y)|=|RL(i,j,x,y)|$, $|LL(i,j,x,y)|\neq 0$, and $|RL(i,j,x,y)|\neq 0$.

   Note that since $W_{i,j}$ is equal to

$$\frac{(|LL(i,j,x,y)|-|RL(i,j,x,y)|)}{\sqrt{2^{MaxHeight}}},$$

   the two situations both result in $W_{i,j}=0$. □

Lemma 2 implies that we can prune the categories that do not contribute to the range sum by checking the corresponding weighting factors. To facilitate the description, we define the *relevant/irrelevant category* as follows:

Definition 6. *The* relevant category *is defined as the category with nonzero weighting factor. The* relevant coefficient *is defined as a coefficient lying in the relevant category. The* irrelevant category/coefficient *is also defined similarly.*

Based on the foregoing, the basic approach, denoted by BASIC, can be clearly described by giving its algorithmic form.

Algorithm BASIC:
Input: the initialized W-$Lists$
Output: top-k streams and corresponding range sums.
1. Neglect all of the coefficients in each category with zero weighting factor
2. Calculate the range sum of all streams by using (3)
3. Find out the k streams with the largest range sum and return

Complexity analysis. The time complexity of the BASIC algorithm is $O(B+M\log M)$, since in step 2, it takes $O(B)$ linear time to calculate the range sums of all streams and, in step 3, it costs $O(M\log M)$ to sort the streams. In particular, when $k \ll M$, the time complexity will be $O(B+kM)$, since tracing M streams k times will be cheaper than sorting the streams.

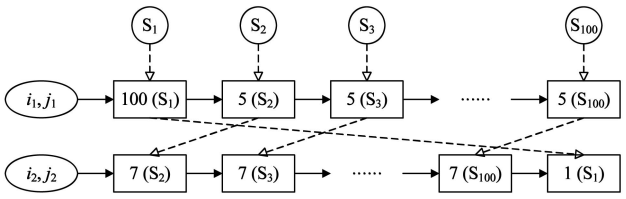


Fig. 10. An illustrative example of searching top-k streams on W-$Lists$.

4.2 PSearch and PAWA

As mentioned above, the BASIC algorithm can retrieve the exact top-k streams by just reading relevant coefficients. In practice, we can further reduce the number of retrieved coefficients if some situations, which prevalently exist in real data, are taken into account. An intuitive example is shown as follows:

Consider 100 streams identified by S_q, $1 \leq q \leq 100$. Suppose that we obtain two relevant categories (i_1,j_1) and (i_2,j_2). For S_1, we have $c(1,i_1,j_1)=100$, and $c(1,i_2,j_2)=1$. As for the other streams S_q, $2 \leq q \leq 100$, we have $c(q,i_1,j_1)=5$ and $c(q,i_2,j_2)=7$. Fig. 10 shows the arrangement of the coefficients, with W-$Lists$ employed. For simplicity, we assume that $W_{i_1,j_1}=W_{i_2,j_2}=1$. To select the top-1 stream, the BASIC algorithm first calculates $\{Sum(q), 1 \leq q \leq 100\}=\{101,12,12,\ldots,12\}$, and then indicates that S_1 has the largest range sum. However, before the answer is found, the BASIC algorithm needs to access all of the coefficients in relevant categories (i_1,j_1) and (i_2,j_2). A more efficient search can be realized by setting the lower bound of the top-1 range sum. Initially, we use the *sorted access* to read the first element, 100 and 7, in each relevant category and set the lower bound to be 107. Note that the lower bound forms the necessary condition of the top-k streams. That is, the range sum of the top-k streams should be larger than or equal to the lower bound. Since 100 and 7 belong to S_1 and S_2, respectively, one of them may have the top-1 range sum. Next, we use the *random access* to read the unknown coefficients of S_1 and S_2 and calculate the range sum of the two streams ($Sum(1)=101$, and $Sum(2)=12$). Since none of the streams satisfies the necessary condition, we proceed to read the next element of each category via the *sorted access* and update the lower bound as 12. The update of the lower bound will make S_1 and S_2 satisfy the necessary condition. We can stop the searching procedure and return the stream with a larger range sum (that is, S_1). Compared to the BASIC algorithm, this searching procedure is more efficient. We use several definitions and lemmas to facilitate the formal description of this concept.

Definition 7. *Given the coefficients that were read via the* sorted access *in the category* (i,j)*, the* minimum weighted coefficient*, denoted by* $Min(i,j)$*, is defined as the coefficient with minimum* weighted value *(that is, the product of the coefficient value and the weighting factor).*

Lemma 3. *The* minimum weighted coefficient *of category* (i,j) *is the latest coefficient that we have read using the sorted access in category* (i,j) *following the path directed by the sign of* $W_{i,j}$ *in* $W\text{-}Lists$.

Proof. It is noted that the coefficients in the category (i,j) are sorted in a descending order. If $W_{i,j} > 0$, following the path provided by the positive sign, the value of the current coefficient is smaller than that of the previous one. The latest coefficient will thus have minimum weighted value among all that we accessed in category (i,j). On the other hand, if $W_{i,j} < 0$, following the path provided by the negative sign, the latest coefficient that we have accessed will lead to maximum weighted value. After multiplying by the weighting factor, the latest coefficient that we have accessed will also lead to the minimum weighted value. □

Given the minimum weighted coefficients for each category, we define the lower bound of the range sum of the top-k streams (denoted by $lbound$) as

$$lbound = \sum_{(i,j)\in \text{ relevant categories}} Min(i,j). \quad (4)$$

The proposed PSearch algorithm will search the coefficients of each category in parallel and validate the candidate streams by comparing the range sum with the lower bound. The PSearch algorithm is outlined in Fig. 11a.

Lemma 4. *The* k *streams returned by the PSearch algorithm are the exact* top-k *streams with maximum range sums.*

Proof. We first note that the candidate streams are sorted according to the range sum in a descending order in heap H. Consider a stream $\hat{s}$ whose ID and range sum are not stored in H. We can find that $W_{i,j}c_{\hat{s},i,j} < Min(i,j)$ for each relevant category (i,j). Therefore, the range sum of each stream $\hat{s}$ not in H is smaller than $lbound$. Since all the other streams are sorted in H according to the range sum, the top-k streams in H are with the k largest range sums. □

Note that we can use the round-robin approach for simplicity to achieve parallelism. According to the property that the coefficients in $W\text{-}Lists$ are sorted in a descending order, the PSearch algorithm decreases the lower bound gradually and examines if the candidate streams satisfy the requirements. The candidate streams are the streams in which there is at least one coefficient larger than or equal to the *minimum weighted coefficient* in a certain relevant category. Briefly, in PSearch, we use the random access to calculate the range sum of the candidate stream and use the sorted access to update the lower bound. If there are at least k candidates, with the range sum larger than or equal to the lower bound, the PSearch algorithm will break without checking the remaining coefficients. Compared to the BASIC algorithm, the PSearch algorithm saves much cost

Algorithm PSearch
Input: the initialized *W-Lists*
Output: top-*k* streams and corresponding range sums
1 Create a heap *H* to store the symbol (*s*,*rsum*), which represents the *id* and the range sum of a specific stream. /*Note that elements in *H* are sorted according to *rsum* in a descending order.*/
2 Perform sorted access in parallel on each relevant category (*i*,*j*), following the path directed by the sign of $W_{i,j}$. **For** each coefficient read by sorted access
3 **if** zero crossing occurs
4 Use random access to find the the range sum (*rsum*) for each stream *u*, which does not contain wavelet coefficient in category (*i*,*j*)
5 Put the symbol (*u*, *rsum*) into *H*
6 Find out the stream *id* (*s*) of this coefficient
7 Use random access to read all the other relevant coefficients of stream *s*
8 Calculate the range sum (*rsum*) of stream *s*
9 Put the symbol (*s*, *rsum*) into *H*
10 Update the lower bound of range sum using *lbound* = $\sum Min(i,j)$
11 **if** there are at least *k* elements with range sum exceeding *lbound*
12 **return** the top-*k* elements and corresponding range sums in *H*.

(a)

Algorithm PAWA
Input: the initialized *W-Lists*
Output: top-*k* streams and corresponding range sums
1 Create a heap *H* for the same purpose as in algorithm PSearch
2 Read the first coefficients of all relevant categories, and calculate *Min*(*i*,*j*) for each category using the weighting factors and these first coefficients.
3 Calculate the lower bound using *lbound* = $\sum Min(i,j)$
4 **while** true
5 Find out the stream *id* (*s*) of the coefficient with max{*Min*(*i*,*j*)}
6 Use random access to read all the other relevant coefficients of stream *s*
7 Calculate the range sum (*rsum*) of stream *s*
8 Put the symbol (*s*,*rsum*) into *H*
9 **if** there are at least *k* elements with range sum exceeding *lbound*
10 **return** the top-*k* elements and corresponding range sums in *H*.
11 **else**
12 Let $(i,j)_{\max}$ = arg max{*Min*(*i*,*j*)} for each relevant category (*i*,*j*)
13 Read the next coefficient of $(i,j)_{\max}$ using sorted access.
14 **if** zero crossing occurs
15 Use random access to find the the range sum (*rsum*) for each stream *u*,
16 which does not contain wavelet coefficient in category (*i*,*j*)
17 Put the symbol (*u*, *rsum*) into *H*
18 Update *Min*(*i*,*j*) of category $(i,j)_{\max}$
19 Update the lower bound of range sum using *lbound* = $\sum Min(i,j)$

(b)

Fig. 11. Algorithmic forms of (a) PSearch and (b) PAWA.

of accessing the coefficients in memory, especially when $M \gg k$. The searching policy of the PSearch algorithm is shown in Fig. 12a. For ease of illustration, we assume that $W_{i,j} > 0$ for each category (i,j). We find that the searching progress in each category is of the same speed.

Since the wavelet coefficients stored in the W-Lists may be smaller than zero, we further address an important effect named "zero crossing." The zero crossing occurs when a negative $Min(i,j)$ is encountered in the category (i,j). Under this circumstance, the data stream s that does not contain the wavelet coefficient in (i,j) should also be considered as a candidate. The reason is explained as follows: If $n(s,i,j) \notin (i,j)$ for stream s, $n(s,i,j)$ will have no contribution during the inverse wavelet reconstruction, indicating that $c(s,i,j) = 0$. After zero crossing, we have $c(s,i,j) > Min(i,j)$. Therefore, the range sums of all the data streams that do not appear in category (i,j) should be calculated.

When PSearch is executed, under many circumstances, the coefficients in some categories may dominate the range sum. For example, the root of a wavelet tree usually plays a more important role than the nodes in the lowest level in

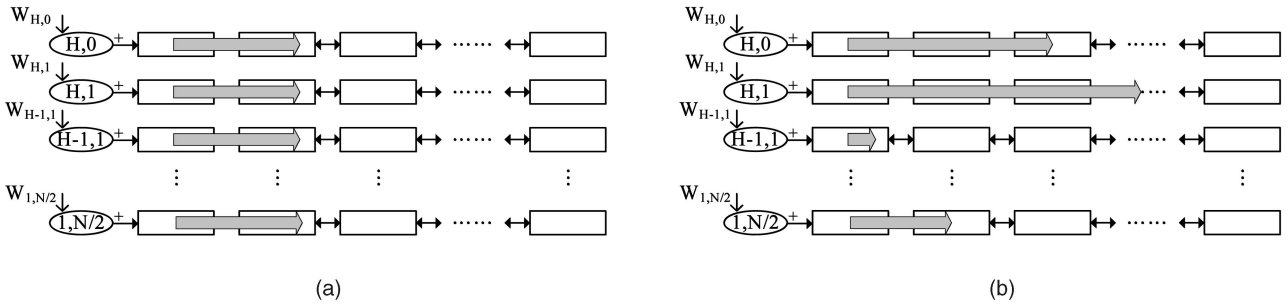


Fig. 12. The policy of the proposed searching approaches. (a) Searching policy of PSearch. (b) Searching policy of the PAWA algorithm.

contributing to the range sum. To address this effect on each coefficient, we quantify the *importance* of each category (i, j) by using the value $Min(i, j)$. The reason is that the coefficients in a more important category tend to have a larger weighted value and thus are more likely to dominate the range sum. Therefore, allowing the category with larger $Min(i, j)$ to have higher priority to be accessed will have better chances to find the exact top-k streams. Moreover, since the importance of each category will change during the search, the algorithm should adjust the *speed* of accessing each category adaptively. According to the characteristics mentioned above, we propose the PAWA algorithm. The algorithmic form of PAWA is outlined in Fig. 11b. Compared to PSearch, the PAWA algorithm saves more access cost due to the adaptive strategy in searching coefficients. The searching policy of the PAWA algorithm is shown in Fig. 12b.

The basic idea of PAWA is inspired by the Threshold Algorithm (TA) proposed in [12]. Similar to the Quick-Combine approach [21], PAWA can be viewed as the adaptive version of TA. It is noted that there are several dissimilarities between PAWA and Quick Combine:

1. For the Quick-Combine algorithm, it is assumed that the score of each object is larger than or equal to zero. However, for our PAWA algorithm, since the wavelet coefficients may be negative, we allow the score of each coefficient to be smaller than zero. For Quick Combine to deal with situations in which some coefficients are negative, it is possible to offset all the values in W-Lists; that is, all coefficients are plus the absolute value of the most negative coefficient. However, since the coefficients in W-Lists may change rapidly as the cells of data streams are received, it is inefficient to offset all the values once the W-Lists is updated.
2. For W-Lists to store the wavelet coefficients, some of the coefficients may not appear in a specific category. This property is different from that of Quick-Combine databases, in which each object will correspond to a score in a specific feature. When the category (i, j) does not contain the coefficient $n(sid, i, j)$, it implies that the value of $n(sid, i, j)$ is zero. As mentioned in item 2a, the Quick-Combine approach can be compatible with the W-Lists structure when all the values in W-Lists are offset by a specific value Δ. However, under this circumstance, the coefficients that are not contained in W-Lists should be assigned a new value Δ after offsetting. Therefore, when Quick Combine is executed, high memory overhead will be incurred to store the offset version of W-Lists.

In brief, if we consider the case of the PAWA algorithm, in which 1) $w(i, j)$ is 1 for each category, 2) each category contains exactly M coefficients, and 3) all coefficients in W-Lists are larger than or equal to zero, the PAWA algorithm will be reduced to the Quick-Combine approach.

4.3 Example of PAWA

According to the synopses at t_8 in the example given in Section 3.2, consider that the query "Select top 1 stream between t_3 and t_7" is processed. Fig. 13a shows how the 16 coefficients are retained in $W\text{-}Lists$ at t_8. Figs. 13b, 13c, and 13d show how the PAWA algorithm accesses the coefficients. Note that in Figs. 13b, 13c, and 13d, we use the shadow to identify a coefficient that is not read by either sorted access or random access. Initially, as shown in Fig. 13b, all coefficients are not accessed, and the weighting factor of each category is initialized according to (2) in response to the query range (3, 7). When the PAWA algorithm starts, as shown in Fig. 13c, the first coefficient of each relevant category (the category with nonzero weighting factor) will be accessed. The lower bound $lbound$ can be obtained according to (4). Note that the coefficient with $Min(i, j)$ in each relevant category is marked by "$*$." Since the coefficient with $\max\{Min(i, j)\}$ belongs to S_1, the PAWA algorithm then uses random access to read all the other coefficients of S_1 (that is, $n(1, 1, 4)$ in category (1, 4)) and inserts the stream identifier and the range sum into the heap H. We find that the range sum of S_1 does not exceed $lbound$. The PAWA algorithm will then proceed to read the next coefficient in category (3, 0), which is the category with $\max\{Min(i, j)\}$, as shown in Fig. 13d. After $lbound$ is updated, the range sum of S_1 exceeds the lower bound. PAWA will return the stream identifier and the range sum of S_1 without accessing all the remaining coefficients.

5 Experimental Evaluation

In this section, we perform extensive experiments to validate each component of the TOMS framework. The simulating environment is presented in Section 5.1. In Section 5.2, we evaluate the performances of different sketching approaches utilized by the synopsis manager. In Section 5.3, we discuss the accuracy for the query processor to answer top-k queries. In Section 5.4, we examine the cost of accessing the wavelet coefficients via

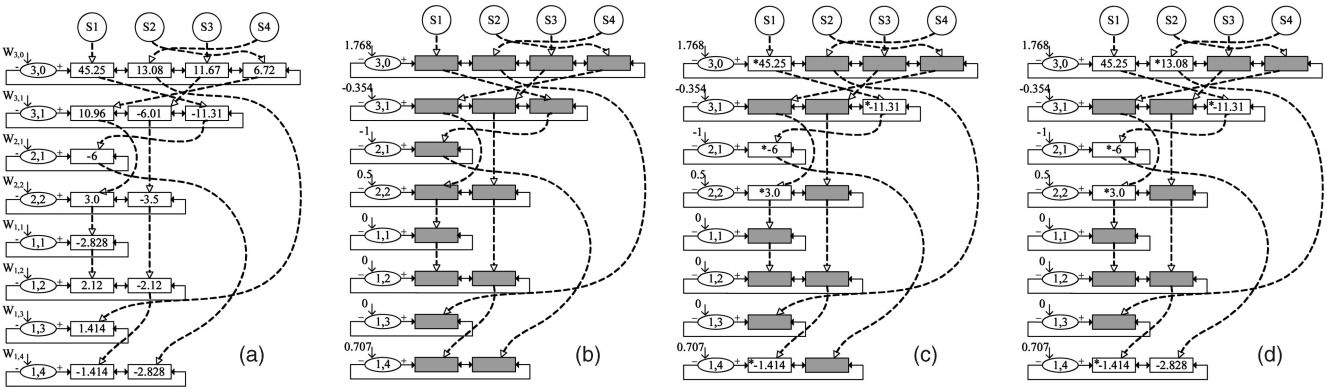


Fig. 13. Execution scenario of the PAWA algorithm. (a) The architecture of the $W\text{-}Lists$. (b) Initialization with the query range (3, 7). (c) Calculate $lbound$ and $rsum$ of S1. (d) Update $lbound$.

different searching strategies. In Section 5.5, remarks of the experiments will be made.

5.1 Simulation Environment

We use real and synthetic data sets to validate the performances of the TOMS framework. The real data set, denoted by STOCK, monitors the stock prices of 234 major companies in Taiwan from 3 January 2005 to 31 March 2005.[7] The stock prices of each company were sampled every 10 seconds.[8] For many data sets in real applications, the change between two neighboring data cells is usually insignificant. Therefore, we also use a simulated data set to inspect the data cells with significant change. The synthetic data set, denoted by SYNTHETIC, contains 100 streams. Each stream corresponds to 32,768 data cells generated by the *random walk* model. It is noted that different streams have different parameter settings in hitting probabilities, barriers, and differences.[9] The reason of selecting different parameters is to ensure that the trends of the data streams will be significantly different. During the experiments, we retain at most B wavelet coefficients of all streams, where B is varied from 10,000 to 50,000. To process the top-k queries, we set the number of entities k to range from 5 to 25. In each query, all of which are uniformly generated, the range defined by the user is of the length $R = 100$. Specifically, let $[sta, end]$ represent the range of the query. sta can be viewed as the random variable uniformly distributed over $[0, N - R)$, and $end = sta + R$.

5.2 Global Error of Reconstruction

Our first experiment examines the global error of approximating multiple streams. Without loss of generality, we choose the L^2-*norm average error* as the error metric. During the experiment, we compare the SYM procedure, which performs online sketching as multiple streams advance, with three approaches: FAIR, STD, and OFFLINE. The FAIR procedure is a sketching procedure, which evenly distributes the B coefficients to every stream. Note that for each stream, the FAIR procedure optimizes the local error metrics shown in Definition 4. The FAIR procedure is viewed as the direct extension from conventional approaches for value-based queries. In the STD approach, which also seeks to minimize the local error, the memory limit of one stream is in proportion to the standard deviation of its data cells.[10] The other approach OFFLINE, which performs offline monitoring by using infinite buffer, calculates the optimal SYM. The experimental results for the SYNTHETIC and STOCK data sets are depicted in Figs. 14a and 14b, respectively.

As shown in Fig. 14, the approximation error decreases as the number of retained coefficients increases. It is reasonable, since more details of the streams can be preserved in the synopses as more wavelet coefficients are retained. We also observe that the performances of the wavelet synopses generated by the SYM procedure are very close to optimal ones. The discrepancy between SYM and OFFLINE is even more implicit at the increase of B. On the other hand, the poor performances of the FAIR procedure indicates that it is actually not fair to assign equal numbers of coefficients to different streams. The reasons are in two aspects. First, since each stream has different trends from other, it is reasonable to assign different numbers of coefficients to different streams. Moreover, the trend of each stream may change as time goes on. The number of coefficients assigned to each stream should also change dynamically to catch different levels of details. As for STD, when there is sufficient memory, this approach may have better performances than FAIR. However, when the memory limit is critical, simply distributing the memory according to the variation without considering the error propagation will lead to an even larger error.

7. In this paper, one of our target applications is to retrieve the information from streams of stock prices. Therefore, we use the data set of stock prices to validate our query processing techniques.

8. It is noted that a 10-second sampling time indicates the format of our STOCK data set rather than the processing time of our SYM procedure. In fact, according to the complexity analysis shown in Section 3, our SYM procedure achieves high efficiency and is capable of processing data cells in high arrival rates.

9. Note that the initial ranking may dominate the answers to the top-k queries in the beginning. However, as the data streams evolve, this effect will be diminished.

10. To calculate the standard deviation, we use the landmark model to achieve adaptiveness. Unlike the window model, the starting point of the landmark model is always fixed to the first data cell. Note that to calculate the standard deviations, we only need to maintain two quantities for each data stream, that is, the sum of data cells and the sum of the squares of the data cells.

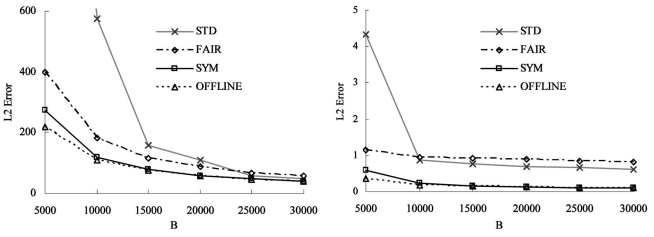


Fig. 14. Error of reconstructing the streams from synopses for SYNTHETIC data and STOCK data. (a) Effect of B for SYNTHETIC data. (b) Effect of B for STOCK data.

In Fig. 14, when B is getting larger, the differences between PAWA and FAIR will become insignificant. An extreme case is that $B = N * M$, which indicates that both the PAWA and FAIR approaches can retain complete information of all data streams. However, in many real-world applications, sufficient memory may not always be available. Therefore, a sophisticated procedure should still be developed to minimize the global error, especially when the size of available memory small. In our experimental results, when the size of available memory is getting smaller, the advantage of SYM over FAIR (or even STD) will become more significant.

5.3 Error of Answering Top-k Queries

We also analyze the error of answering the top-k queries. Since the wavelet coefficients retained in memory only provide an approximation, reconstruction from these synopses may change the rank of these streams within a certain range. These errors degrade the accuracy of answering the top-k queries. Since the query processor can find exact top-k streams, the accuracy of answering the top-k queries depends on the sketching techniques employed in the synopsis manager. To discuss the reliability of the query processor, we compare the sketching techniques SYM, FAIR, STD, and OFFLINE mentioned in Section 5.2. We define two error metrics as follows: The *non-rank-based error* is defined as the probability that the query processor returns an incorrect answer to the query. An incorrect answer means that not all of the top-k streams indicated by the query processor have the k largest range sums. The *rank-based error* is defined as the probability that the top-k streams indicated by the query processor have the wrong rank. It is noted that the top-k answer may still be correct, even though the k streams have the wrong rank. We also measure the average recall of processing the top-k queries. Overall, the rank-based error metric is tighter than the non-rank-based error metric, and the non-rank-based error metric is tighter than the recall metric. We use the following examples to illustrate the tightness of the three error metrics.

Example 2. Consider that the correct top-4 answer are $\{S_0 > S_1 > S_2 > S_3\}$, where $S_i > S_j$ indicates that the rank of S_i is higher than S_j. If the query processor returns $\{S_0 > S_1 > S_2 > S_4\}$, the value of recall will be 0.75. For rank-based error and nonrank-based error, the answer is a "wrong" answer.

Example 3. Consider the correct top-4 answer

$$\{S_0 > S_1 > S_2 > S_3\}.$$

If the query processor returns $\{S_0 > S_1 > S_3 > S_2\}$, the value of recall will be 1. For the non-rank-based error metric, it is still a correct answer. However, for the rank-based error, it is a "wrong" answer, since S_2 and S_3 are in wrong ranks.

Figs. 15 and 16 depict the error metrics induced by different sketching procedures, with B and k varied after 5,000 user queries are processed. In each query, the range defined by the user is of the length $R = 100$. From the experimental results, the OFFLINE procedure will generate optimal synopses. Note that OFFLINE only guarantees that the L^2-norm average error can be minimized, which means that at some data points, there may be still significant errors. Since the significant errors may change the rank of the data streams within a specific range, OFFLINE may lead to the errors of answering top-k queries. On the other hand, due to the fact that OFFLINE, SYM, STD, and FAIR employ different policies in discarding the wavelet coefficients, the significant errors will occur at different positions. Since the users may query the top-k range sums within arbitrary ranges, it is still possible that in certain queries, the OFFLINE procedure leads to significant errors, whereas the other approaches lead to insignificant errors. Specifically, the OFFLINE approach leads to the minimum error ranks on the average. Therefore, this phenomenon will become negligible when large numbers of query processing results are collected.

As shown in Fig. 15, the *non-rank-based error* and the *rank-based error* diminish as the parameter B increases.[11] This result agrees with our intuition that retaining more coefficients enables each stream to have accurate range sum. On the other hand, the *rank-based error* of each approach is larger than the *non-rank-based error*, given the same parameter settings. This observation can also be explained by the tighter restriction of the *rank-based error* measure. Since the SYM procedure can approximate the offline wavelet synopses very accurately from previous experiments, we can find that the synopses generated by the SYM procedure will lead to high accuracy of answering top-k queries. As for the recall metric, since it is the loosest metric, SYM and OFFLINE can achieve more than 96 percent of recalls. However, the performances of FAIR and STD degrade drastically as B decreases. Similarly, in Fig. 16, we find that the SYM procedure outperforms the FAIR procedure. Fig. 16 also shows that the SYM-based synopsis manager will lead to accurate answers to the top-k queries when k is small. As k increases, the reliability of the query processor degrades, even though the offline sketching approach is employed. This experiment shows that a well-designed synopsis manager will enhance the reliability of the query processor.

11. Although the OFFLINE approach guarantees that the L^2-norm average errors are minimized, the ranks of the data streams are still not guaranteed. Therefore, as we increase B, some of the top-k streams may become outliers. In the long term, we can still observe that increasing B is helpful to reducing the nonrank-based error.

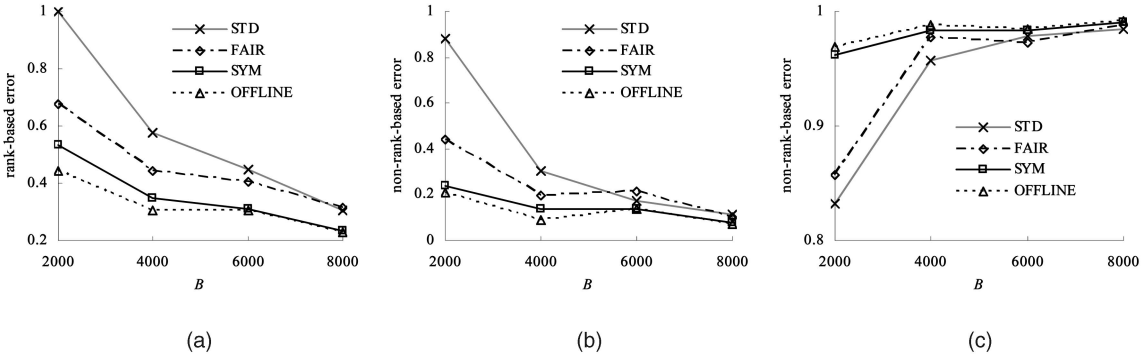


Fig. 15. Error of answering top-k queries, with B varied in SYNTHETIC data sets $(R = 100, k = 10)$. (a) Rank-based error. (b) Non-rank-based error. (c) Recall.

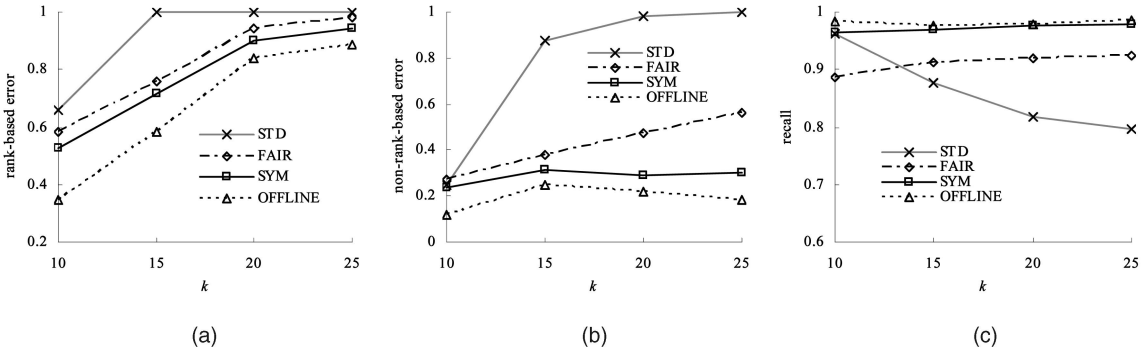


Fig. 16. Error of answering top-k queries, with k varied in SYNTHETIC data sets $(B = 2,000, R = 100)$. (a) Rank-based error. (b) Non-rank-based error. (c) Recall.

To evaluate the reliability of SYM, we consider Figs. 15b and 16b. In many applications, the users may be satisfied when the query processor returns correct top-k entities without order. This effect can be reflected by the non-rank-based error metric. In Fig. 15b, when the memory limit is critical, for SYM, about 80 percent of the answers are still correct. However, the reliability of FAIR is reduced to about 50 percent. On the other hand, in Fig. 16b, the SYM approach can still maintain its reliability as more top-ranked entities are queried, whereas the other two approaches FAIR and STD suffer from larger error. Therefore, for critical situations such as insufficient memory and huge queried entities, we suggest that SYM should be employed when monitoring multiple streams.

5.4 Efficiency Analysis

Next, we discuss the efficiency issue of the PAWA algorithm. To evaluate the efficiency, we measure two performance metrics: cost of memory access and execution time. For the cost of memory access, we calculate the number of accessed coefficients (denoted by NOA) of each query by averaging 1,000 different user queries, which have been processed in the query processor. In addition to evaluating the performances of BASIC, PSearch, and PAWA, we also calculate the optimal memory access for comparison purposes. The cost of optimal access is obtained by examining all possible strategies exhaustively, given that the top-k streams are known in advance. As for the execution time, we aggregate the processing time of these 1,000 queries. Note that OPT indicates the lower bound and cannot be achieved by any approach. Therefore, we do not consider the OPT method when measuring the execution time. In Fig. 17, we vary the number of retained coefficients and evaluate the quality of each algorithm. As for Fig. 18, the number of accessed coefficients is measured, with the number of queried entities varied.

In Fig. 17a, we observe that more coefficients are accessed for each algorithm at the increase of B. The reason is that there are more categories in the $W\text{-}Lists$ to represent more details of the streams as the total number of retained coefficients increases. Therefore, more coefficients should be checked before the top-k entities are obtained. Among all approaches, the BASIC algorithm has the poorest quality, since all of the coefficients in the relevant categories must be accessed. Compared to BASIC, the PSearch algorithm only needs to access about half of the coefficients in the relevant categories. As for PAWA, this algorithm has the most outstanding perfor-

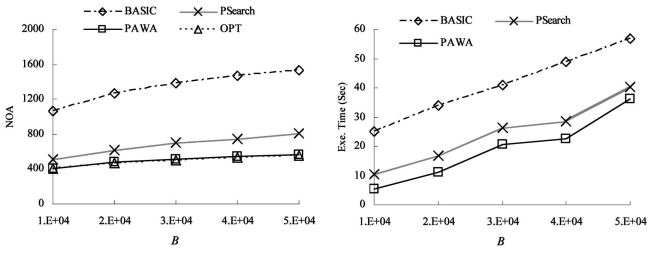


Fig. 17. Efficiency of the PAWA algorithm in SYNTHETIC data sets, with B varied. (a) Cost of memory access. (b) Execution time.

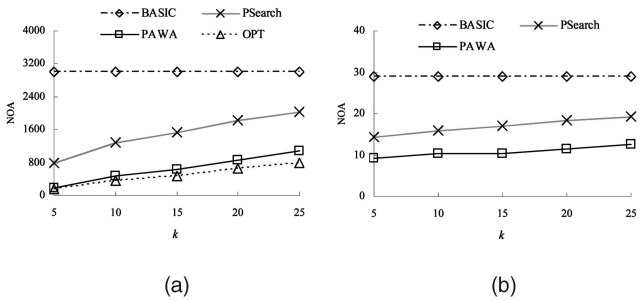


Fig. 18. Efficiency of the PAWA algorithm in STOCK data sets, with k varied. (a) Cost of memory access. (b) Execution time.

mances due to the adaptive searching strategy. Moreover, the quality of the PAWA algorithm is very close to the optimum. This phenomenon is even more distinguishable for the SYNTHETIC data set. The reason is that due to different initial values and different step sizes, some data streams will have much higher values than others within a certain range. Therefore, the coefficients in a small part of categories may dominate the rank of the data streams. When PAWA is executed, the coefficients that tend to dominate the range sum will have higher priority to be accessed. This property will make PAWA achieve very similar performances to the optimal policy in accessing the coefficients. It is also noted that the memory access cost of the PAWA algorithm is very insensitive to the variation of B. This phenomenon indicates that the advantage of the PAWA algorithm over the other two will be more prominent when larger numbers of wavelet coefficients are kept.

On the other hand, as we inspect the execution time in Fig. 17b, we observe that the discrepancies are different from those in Fig. 17a. The major reason is that the execution time reflects the effect of some operations, which may depend on the system platform. For example, for PSearch and PAWA, it may take some execution time to examine whether a data stream contains the wavelet coefficient in a specific category during the random access. Moreover, when PAWA is executed, the operation of finding the data stream with $\max\{Min(i,j)\}$ also influences the execution time.

As depicted in Fig. 18, it can be seen that the PAWA algorithm outperforms the other two as the queries for different numbers of top-k entities are processed. For the BASIC algorithm, since all of the coefficients in relevant categories are accessed, the performances are independent of parameter k. For PSearch and PAWA, although these two algorithms only access part of the coefficients in the relevant categories, a larger k will increase the searching cost, since more sorted accesses are required before the appropriate lower bound is obtained. This result implies that the PAWA and PSearch algorithms are very suitable for processing top-k queries under the circumstance of $k \ll M$.

5.5 Remarks

From the experimental results, the conventional monitoring approaches, which aim at minimizing the local error metrics for each individual stream, will degrade the reliability of query processor in some critical situations. On the other hand, the SYM-based synopsis manager, which minimizes the global error metrics, can provide accurate synopses and thus enhance the reliability of answering the top-k queries. This phenomenon justifies the necessity of optimizing the synopsis manager. Moreover, after evaluating the performances of all searching strategies employed by the query processor, we observe that the PAWA algorithm achieves near-optimal quality. In order to process the top-k queries more efficiently, we suggest that the PAWA algorithm should be adopted by the query processor. Therefore, our TOMS framework, which optimizes the synopsis manager and the query processor by employing the SYM procedure and PAWA algorithm, is best suited to process top-k queries over multiple streams.

6 Conclusion

In this paper, we proposed a general framework TOMS to maintain the wavelet coefficients of multiple streams and to process top-k queries according to the wavelet synopses. We optimized the synopsis manager for multiple streams by using the SYM procedure so that the predetermined global error metric can be minimized. We retain the associated wavelet coefficients by using a novel data structure $W\text{-}Lists$, which can exactly respond to the interested range defined by the user. To process the top-k query in the query processor, we proposed the PAWA algorithm to calculate the accurate answer with minimized memory access. The experimental results show that our framework can process top-k queries accurately and efficiently.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan, ROC, under Contracts NSC93-2752-E-002-006-PAE.

References

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and Issues in Data Stream Systems," *Proc. 21st ACM Symp. Principles of Database Systems (PODS '02)*, 2002.

[2] B. Babcock and C. Olston, "Distributed Top-k Monitoring," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03)*, 2003.

[3] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden, "Progressive Distributed Top-k Retrieval in Peer-to-Peer Networks," *Proc. 21st IEEE Int'l Conf. Data Eng. (ICDE '05)*, 2005.

[4] M. Bawa, R.J. Bayardo, S. Rajagopalan, and E.J. Shekita, "Make It Fresh, Make It Quick—Searching a Network of Personal Webservers," *Proc. 12th Int'l World Wide Web Conf. (WWW '03)*, 2003.

[5] A. Bulut and A. Singh, "SWAT: Hierarchical Stream Summarization in Large Networks," *Proc. 19th IEEE Int'l Conf. Data Eng. (ICDE '03)*, 2003.

[6] A. Bulut and A. Singh, "A Unified Framework for Monitoring Data Streams in Real Time," *Proc. 21st IEEE Int'l Conf. Data Eng. (ICDE '05)*, 2005.

[7] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases," *ACM Trans. Database Systems*, vol. 27, no. 2, 2002.

[8] S. Chaudhuri, L. Gravano, and A. Marian, "Optimizing Top-k Selection Queries over Multimedia Repositories," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, no. 8, Aug. 2004.

[9] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Evaluating Probabilistic Queries over Imprecise Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03)*, 2003.

[10] R. Chengy, B. Kaox, S. Prabhakarz, A. Kwanx, and Y. Tuz, "Adaptive Stream Filters for Entity-Based Queries with Non-Value Tolerance," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05)*, 2005.
[11] G. Cormode, S. Muthukrishnan, and I. Rozenbaum, "Summarizing and Mining Inverse Distributions on Data Streams via Dynamic Inverse Sampling," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05)*, 2005.
[12] R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware," *Proc. 20th ACM Symp. Principles of Database Systems (PODS '01)*, 2001.
[13] M. Garofalakis and P.B. Gibbons, "Probabilistic Wavelet Synopses," *ACM Trans. Database Systems*, vol. 29, no. 1, 2004.
[14] M. Garofalakis and A. Kumar, "Deterministic Wavelet Thresholding for Maximum-Error Metrics," *Proc. 23rd ACM Symp. Principles of Database Systems (PODS '04)*, 2004.
[15] A.C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M.J. Strauss, "One-Pass Wavelet Decompositions of Data Streams," *IEEE Trans. Knowledge and Data Eng.*, vol. 15, no. 3, May/June 2003.
[16] S. Guha, D. Gunopulos, and N. Koudas, "Correlating Synchronous and Asynchronous Data Streams," *Proc. Ninth ACM Int'l Conf. Knowledge Discovery and Data Mining (KDD '03)*, 2003.
[17] S. Guha and B. Harb, "Wavelet Synopsis for Data Streams: Minimizing Non-Euclidean Error," *Proc. 11th Int'l Conf. Knowledge Discovery and Data Mining (KDD '05)*, 2005.
[18] S. Guha, C. Kim, and K. Shim, "XWAVE: Approximate Extended Wavelets for Streaming Data," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB '04)*, 2004.
[19] S. Guha, N. Koudas, and K. Shim, "Approximation and Streaming Algorithms for Histogram Construction Problems," *ACM Trans. Database Systems*, vol. 31, no. 1, 2006.
[20] S. Guha, K. Shim, and J. Woo, "REHIST: Relative Error Histogram Construction Algorithms," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB '04)*, 2004.
[21] U. Güntzer, W.-T. Balke, and W. Kießling, "Optimizing Multi-Feature Queries for Image Databases," *Proc. 26th Int'l Conf. Very Large Data Bases (VLDB '00)*, 2000.
[22] M.J. Hsieh, M.S. Chen, and P.S. Yu, "Integrating DCT and DWT for Approximating Cube Streams," *Proc. 14th ACM Int'l Conf. Information and Knowledge Management (CIKM '05)*, 2005.
[23] H.V. Jagadish, H. Jin, B.C. Ooi, and K.-L. Tan, "Global Optimization of Histograms," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '01)*, 2001.
[24] P. Karras and N. Mamoulis, "One-Pass Wavelet Synopses for Maximum-Error Metrics," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05)*, 2005.
[25] N. Koudas, B.C. Ooi, K.-L. Tan, and R. Zhang, "Approximate NN Queries on Streams with Guaranteed Error/Performance Bounds," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB '04)*, 2004.
[26] K.H. Liu, W.G. Teng, and M.S. Chen, "Incremental Maintenance of Wavelet Synopses for Data Streams," *Proc. ICDM Workshop Temporal Data Mining: Algorithms, Theory and Applications (TDM '05)*, 2005.
[27] Y. Matias, J.S. Vitter, and M. Wang, "Wavelet-Based Histograms for Selectivity Estimation," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '98)*, 1998.
[28] S. Michel, P. Triantafillou, and G. Weikum, "KLEE: A Framework for Distributed Top-k Query Algorithms," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05)*, 2005.
[29] S. Papadimitriou, J. Sun, and C. Faloutsos, "Streaming Pattern Discovery in Multiple Time Series," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05)*, 2005.
[30] C.S. Perng, H. Wang, S.R. Zhang, and D.S. Parker, "Landmarks: A New Model for Similarity-Based Pattern Querying in Time Series Databases," *Proc. 16th IEEE Int'l Conf. Data Eng. (ICDE '00)*, 2000.
[31] E.J. Stollnitz, T.D. Derose, and D.H. Salesin, *Wavelets for Computer Graphics: Theory and Application.* Morgan Kaufmann, 1996.
[32] 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)*, 1999.

Hao-Ping Hung received the BS degree from the Department of Electrical Engineering, National Taiwan University, Taipei, in 2001, and the PhD degree from the Graduate Institute of Communication Engineering, National Taiwan University, in January 2007. He is currently a senior engineer at CyberLink Corp. His research interests include mobile computing, resource allocation in the wireless environment, multimedia networking, and data streams.

Kun-Ta Chuang received the BS degree from the National Taiwan Normal University, Taipei, in 2000 and the PhD degree in communication engineering from the National Taiwan University, Taipei, in 2006. He is currently a software engineer at SYNOPSYS Inc., developing physical verification tools. His research interests include data mining, mobile data management, and electronic design automation. He is a member of the IEEE.

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 a research staff member at the IBM T.J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996. In addition to serving as the program chair/vice-chair and a keynote/tutorial speaker in many international conferences, he was an associate editor of the *IEEE Transactions on Knowledge and Data Engineering* and also the *Journal of Information Science and Engineering*, is currently on the editorial board of the *Very Large Data Base (VLDB) Journal*, *Knowledge and Information Systems*, and the *International Journal of Electrical Engineering*, and is a distinguished visitor of IEEE Computer Society for Asia-Pacific from 1998 to 2000 and also from 2005 to 2007. He is currently a distinguished professor jointly appointed by the Electrical Engineering Department, the Computer Science and Information Engineering Department, and also the Graduate Institute of Communication Engineering, National Taiwan University. His research interests include database systems, data mining, mobile computing systems, and multimedia networking. He has published more than 240 papers and holds, or has applied for, 18 US patents and 7 ROC patents 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, and also the Outstanding Innovation Award from IBM Corporate for his contribution to a major database product. He also received numerous awards for his research, teaching, inventions, and patent applications. He is a fellow of the ACM and the IEEE.