SIMILARITY SEARCH IN RECENT BIASED TIME SERIES DATABASES

Muruga Radha Devi D.[1], **Thambidurai P.**[2],

[1]Research Scholar, Sathyabama University, Chennai, India
[2]Department of CSE, PKIET, Karaikal, India Email: [1] emrdevi@hotmail.com

ABSTRACT

A time series database is a collection of data that are generated in series as time goes on and constitutes a large portion of data stored in computers like stock-price movements, weather data, bio-medical measurements, video data etc., Two time sequences of same length are said to be similar if the Euclidean distance is less or equal to a given threshold. The main issue of similarity search in time series databases is to improve the search performance since time sequences are usually of high dimension. So it is important to reduce the search space for efficient processing of similarity search in large time series databases. We have used Adaptive Framework for the data reduction purpose which improves the search performance in Recent-Biased time series databases. We have applied a set of linear transformations on the reduced sequence that can be used as the basis for similarity queries on time series data. We have also formalized the intuitive notions of exact and approximate similarity in time series data. Our experiments show that the performance of this method is competitive to that of processing ordinary queries using the index and much faster than sequential scanning.

Key words: Similarity search, Data Reduction, Recent-Biased Time series, Adaptive Framework, Transformations.

I. INTRODUCTION

Time series data are of growing importance in many new database applications such as data mining and data warehousing. A time series is a sequence of real numbers each number representing a value at a time point. For example the sequence could represent stock or commodity prices, sales exchange rates, weather data, bio-medical measurements. Time series are typically huge as well as of high dimension . Therefore time series analysis has been an important and challenging topic in machine learning and data mining.

Recently there has been much interest in the problem of similarity search in time series databases. For e.g. we may want to find the stocks that behave in approximately the same way or stocks that increased linearly upto sometime and then crashed; or years when the temperature patterns in two regions of the world were similar. In these type of queries approximate rather than exact matching is required.

The similarity between two time series is typically measured with Euclidean distance which can be calculated very efficiently. However the volume of data encountered exasperates the problem. Consider the MACHO project where database contains more than a terabyte of data and is updated at the rate of several GB a day.

Applications in forecasting involve predicting the future conditions using the last few measurements. For example in the case of banner-hits data, the number of hits in the immediate past can be used to gauge the popularity of an advertisement. In fraud and security monitoring, the recent data has more predictive value compared to the old data. Therefore a system which maintains better approximations for the recent data is useful. The challenge is to maintain these biased approximations continuously as new data arrives in an online manner. *So the most promising similarity search methods perform reduction on the data and then use an index structure for the data in the transformed space.*

The widely used data reduction techniques are SVD, DFT, DWT, PIP, PAA etc., These data reduction strategies are specifically developed for time series analysis in general. However they are non-actionable in recent-biased analysis for streaming time series. This is because Traditional Time series analysis algorithms take recent data and old data as equally important. In recent-biased analysis, recent data are much more important than old ones. Most existing data reduction techniques process time series in a batch way(i.e.) the whole time series needs to be examined again on the arrival of new data. So they are very inefficient for processing online data streams.

This leaves a big gap between the recent-biased analysis and traditional machinery available to data reduction. To bridge this gap an adaptive framework is been used for recent-biased analysis over online data streams. In this method the time series is partitioned into segments and then data reduction technique is applied to each segment where more coefficients are kept for more recent data while fewer are kept for the older data. Thus, more details are preserved for recent data and fewer coefficients are kept for the whole time series which improves the efficiency greatly.

Our Contribution

First our idea is to apply a data reduction technique to different parts of time series and then more coefficients are kept for more recent segments with fewer coefficients for older segments referred as **Adaptive Framework** for recent biased time series where vari-DWT are employed efficiently to obtain compact synopses of general relational tables.

Second we use a class of transformations that can be used in a query language to express similarity in a fairly general way.

Third we provide syntax and semantics for similarity queries that account for approximate matching, scaling and shifting that have efficient indexing support.

II. BACKGROUND AND RELATED WORK

In this section, research into various similarity search methods are first reviewed and then we briefly describe the recent biased time series analysis and about the traditional data reduction methods.

A. Similarity Search

The presence of time component in data is what unifies such diverse datasets and classifies them as time series. So much research has been devoted recently to the efficient management of time-series data. Analysis of time-series data is rooted in the ability to find similar series.

Similarity is defined in terms of a distance metric, most often Euclidean distance or relatives of Euclidean distance. Given two time series $Q = (q_1, q_2, \dots q_n)$ AND $C = (c_1, c_2, \dots c_n)$, their Euclidean distance is defined as

$$D(Q, C) = \sqrt{\Sigma (q_i - c_i)^2} \qquad [1]$$

where $i = 1, 2 \dots n$

Two time sequences of same length are said to be similar if the Euclidean distance is less or equal to a given threshold.

Agrawal, Faloutsos and swami [1], first proposed the use of distance preserving transformation for dimensionality reduction. The transformations are applied to the original data and a few coefficients of the transformed data are then indexed. Queries on the data are transformed into queries on these features that can efficiently be answered using the index. The answer in the feature space, when converted back to the data space, must be a superset of the original query answer.

There are two ways in which the data is organized for similarity search namely:

1. Whole matching: Here all sequences to be compared are of the same length
2. Subsequence matching: Here we have a query sequence Q of length n and a longer sequence C of length m. The task is to find the subsequence in C of length n, beginning at C_i, which best matches Q and report its offset within C.

Whole matching requires comparing the query sequence to each candidate sequence by evaluating the distance function and keeping track of the sequence with the lowest distance. Subsequence matching requires that the query Q be placed at every possible offset within the longer sequence.

There are two types of queries that we would like to support in time series database namely range queries and nearest neighbor query. Range query returns all sequences within an epsilon of the query sequence whereas nearest neighbour query returns k-nearest sequence to the query sequence.[5] The brute force approach to answering these queries is sequential scanning which requires comparing every time series C_i to Q_i. This approach is unrealistic for large datasets. Similarity searching techniques that guarantee no false dismissals are exact, and techniques that do not have guarantee as approximate. Approximate techniques can still be very useful for exploring large databases, when the probability of false dismissal is low [9].

Approximate Techniques for Similarity Searching

Transforming the data with a lossy compression schemes, then doing a sequential search on the compressed data. Existing techniques [9], suffer from some limitation. (i.e.) they are all evaluated on small datasets residing in main memory, and it is unclear if they can be made to scale to large datasets.

Exact Techniques for Similarity Searching

A time series C = $(c_1, c_{2,\ldots\ldots} c_n)$ with n data points can be considered as a point in n-dimensional space which suggests that time series could be indexed by multidimensional index structure such as R-tree[8], so we need to perform dimensionality reduction in order to exploit multidimensional index structure to index time series data. In order to guarantee no false dismissals the distance measure in the index space must satisfy the following condition

$$D_{\text{indexspace}}(A, B) \leq D_{\text{true}}(A, B) \qquad [2]$$

This is known as Contractive Property.

Rafiei and Mendelzon [12] propose a set of linear transformations such as moving average, time warping and reversing. These transformations can be used as the basis of similarity queries for time series data. In addition Rafiei[13] propose the method of processing queries that express similarity in terms of multiple transformations instead of a single one. B-K, Jagadish and Faloutsos [2] used time warping as distance function and present algorithms for retrieving similar time sequences under this function. However a time warping distance does not satisfy triangular inequality and can cause false dismissals.

B. Recent-Biased Time Series Analysis

In many applications, such as the stock market, we care more about the recent data than what happened long ago. Besides the global trend, the recent data are very important for us to judge the similarity between time series and more significant for us to predict and make decisions than the details of ancient data. For example, for a stock broker, the long-term trend of stock price and the Tree) to process queries over data streams, that are biased towards the more recent values. Palpanas et al [11] have proposed a technique for online amnesic approximation for streaming time series and the proposed representation of time series can represent arbitrary user-specified amnesic functions.

C. Popular Techniques for Data Reduction

Because of the high dimensionality of most time series, the direct indexing of time series is prohibitive. As a result **Data Reduction** appears to be the most promising method for overcoming this problem. SVD(Singular Value Decomposition) or PCA(Principal Component Analysis)[4] is a classic technique for dimension reduction which keeps the Euclidean distance between the time series with singular values. SVD is a global transformation technique (i.e.) the entire data set is examined to compute the axes. Popular feature extraction techniques are DFT(Discrete Fourier Transform) and DWT(Discrete Wavelet Transform) where the sequence is projected into the frequency domain(DFT) or tiling of the time frequency plane(DWT) [3]. Keogh et al and Faloutsos et al[6] independently suggested approximating a time series by dividing it into equal length segments and recording mean value of the data points that fall within the segment. The authors call it Piecewise Aggregate Approximation which is more competitive with more sophisticated transforms. If we allow the segments to have arbitrary lengths, we call that as Adaptive Piecewise Constant Approximation[7] which requires two numbers per segment, the first number records the mean value of all the data points in the segment, the second number records the length.

III. ADAPTIVE FRAMEWORK FOR DATA REDUCTION

In stream data analysis, users usually pay more attention to recent data than to old data, and are often interested in recent changes at a fine scale, but long-term variations at a coarse level. So it is reasonable to process time series data with bias on recent values. [14] With bias on recent data, space requirement will be much reduced and the index and query on time series will be more efficient.

Adaptive Data Reduction is to reduce the time series data while minimizing the distance between the original time series and the reconstructed time series.

A. Steps Involved in Adaptive Data Reduction

(i) Time series is divided into segments, where recent data are partitioned into smaller segments to keep more details and larger segments can be set for older data, so that less detail is kept for them. The size of the segments is set exponentially because it is more space efficient and DWT run fastest when the length of time series is the power of two. Therefore the length of segment S is set to $n_i = 2$ for $i = 1, 2, 3 \ldots n$ or it can be set to any monotonously increasing number sequences.

(ii) After partitioning a data reduction technique is applied to each segment and the same number of coefficients are kept for every segment.

(iii) If the coefficient number for a segment is less than the coefficient number k, then all available coefficients are kept for that segment. Since older data are partitioned into larger segments, they are kept at a coarser scale than more recent data. It is very easy for this scheme to process online data streams.

(iv) When a new value arrives, a new segment is generated for it. Then the latest pair of neighbouring segments that are of the same length, are merged into a single segment, At most one pair of segments are merged when a new value is added.

(v) Only the new segment generated by merging needs to be processed again with data reduction techniques, with other segments intact. (i.e) only a small part of the whole time series needs to be re-examined.

Assume the length of segment S_i to be 2^i (i.e.) the length of segments S3, S2, S1 are 4, 2 and 1 from old data to recent data and almost eight coefficients are kept for almost all segments. When a new data arrives, a new segment S0 is created for it, so the sizes of segments are 4, 2, 1 and 1. Since the last two segments are of same length they are merged while the other two segments remains unchanged. Therefore there are 3 segments left whose sizes are 4, 2 and 2. When another data arrive, the size of the segments become 4,2, 2 and 1 so the two segments in the middle are merged and the sizes of segments become 4, 4 and 1. If another data arrives the sizes of segments become 4, 4, 1 and 1 so last two segments are merged to have 4, 4, and 2 where the first two segments are not merged because at a time only one pair of segments are merged when a new value arrives. In the above procedure almost one merge happens at each step. Only the new segment generated by merging needs to be processed again with data reduction techniques.

B. Wavelet Decomposition

Wavelet Decomposition of a function consists of a coarse overall approximation together with detail coefficients that influence the function at various scales. Haar wavelets are conceptually simple and very fast to compute and have been found to perform well in practice for a variety of applications ranging from image editing and querying.

Suppose we are given a one-dimensional data vector containing the following eight values $A = [2, 2, 0, 2, 3.5, 4, 4]$, the Haar wavelet transform of A can be computed as follows. We first average the values together pair-wise to get a new "lower-resolution" representation of the data with the following average values. In other words, the average of the first two values (that is, 2 and 2) is 2 and that of the next two values (that is, 0 and 2) is 1. Obviously, some information has been lost in this averaging process. To be able to restore the original values of the data array, we need to store some ***detail coefficients***, that capture the missing information. In Haar wavelets, these detail coefficients are simply the differences of the (second of the) averaged values from the computed pair-wise average.

Table 1. Wavelet Decomposition

Resolution	Averages	Detail Coefficients
8	(2, 2, 0, 2, 3, 5, 4, 4)	–
4	(2, 1, 4, 4)	(0, – 1, – 1, 0)
2	(1.5, 4)	(0.5, 0)
1	(2.75)	(–1.25)

We define the ***wavelet transform*** (also known as the ***wavelet decomposition***) of *A* to be 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 $W_A = \{2.75, -1.25, 0.5, 0, 0, -1, -1, 0)$ Each entry in W_A is called a ***wavelet coefficient***. The main advantage of using W_A instead of the original data vector *A* is that for vectors containing similar values most of the detail coefficients tend to have very small values. Thus, eliminating such small coefficients from the wavelet transform (i.e., treating them as zeros) introduces only small errors when reconstructing the original data, giving a very effective form of lossy data compression.

The wavelet decomposition is very efficient computationally, requiring only $O(N)$ CPU time and $O(N/B)$ *I/Os* to compute for a signal of N values. No information has been gained or lost by this process. The original signal has eight values and so does the transform. Given the transform, we can reconstruct the exact signal by recursively adding and subtracting the detail coefficients from the next lower resolution. For compression reasons, the detail coefficients at each level of recursion are often normalized; the coefficients at the lower resolutions are weighted more heavily than the coefficients at the higher resolutions. One advantage of normalized wavelet transform is that in many cases a large number of detail coefficients turn out to be very small in magnitude. Truncating these small coefficients from the representation introduces only small errors in the reconstructed signal. We can approximate the original signal effectively by keeping the most significant coefficients.

C. Processing Relational Queries in the Wavelet-Coefficient Domain

In this section, we propose a novel query processing method for wavelet-coefficient synopses. The basic operators of our method correspond directly to conventional relational algebra and SQL operators, including the (non-aggregate) select, project, and join, as well as aggregate operators like count, sum, and average. There is, however, one crucial difference: our operators are defined **over the wavelet-coefficient domain**, that is, their input(s) and output are *sets of wavelet coefficients* (rather than relational tables). The motivation for defining a query processing method for wavelet coefficients comes directly from the need for efficient approximate query processing.

To see this, consider an n-ary relational query $\hat{f}$ over $R1 \ldots Rn$ and assume that each relation Ri has been reduced to a (truncated) set of wavelet coefficients W_{Ri}. The synopsis W_{Ri} is a highly compressed representation of render(W_{Ri}) that is typically orders of magnitude smaller than Ri. Executing $\hat{f}$ in the compressed wavelet-coefficient domain can offer tremendous speedups in query execution cost. We therefore define the operators op of our query processing method over wavelet-coefficient synopses, while guaranteeing the valid semantics depicted pictorially in the transition diagram of Figure 1. (These semantics can be translated to the equivalence render(op(W_{Ri})) op(render(W_{Ri})), for each operator op.) Our method allows the fast execution of any relational query *entirely* over the wavelet-coefficient domain, while guaranteeing that the final result is identical to that obtained by executing $\hat{f}$ on the approximate input relations.

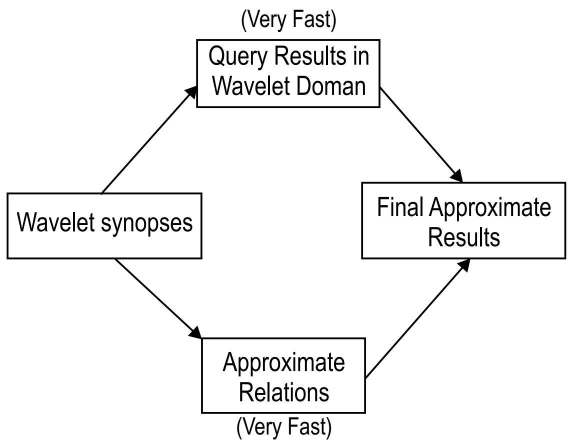

Fig. 1. Valid semantics for processing query operators over the wavelet-coefficient domain.

IV. SIMILARITY SEARCH METHODS

The most basic problem in similarity search is First-Occurrence Subsequence Matching which is defined as follows: Given a query sequence Q of length n and a longer sequence C of length N, find the first occurrence of a contiguous subsequence within C that matches Q exactly. *A* variant of the above problem involves finding all occurrences which is called as All occurrences Subsequence Matching problem. The above two problems have variants when the data consists of many sequences of same length as the query. The All occurrences whole sequence Matching problem is given a query sequence Q of length n and a set of N/n data sequences, all of the same length find all of the data sequences that match Q exactly.

Time series data in continuous domain is inherently inexact, due to the unavoidable imprecision of measuring devices and clocking strategies. Given a tolerance $\varepsilon > 0$ and a distance metric D between sequences, sequence $S1$ and $S2$ match approximately within tolerance ε when $D(S1, S2) \leq \varepsilon$.

A. Transformations on Time Series Data

In many cases, it is more natural to allow the matching of sequences that are not close to each other in an Euclidean distance. For example, two companies may have identical stock price fluctuations, but one's stock is worth twice as much as the other at all times. For another example, two sales patterns might be similar even if the sales volumes are different. We shall refer to the difference between these sequences as **scale**. In another example, the temperature on two different days may start at different values but then go up and down in exactly the same way. We shall refer to the difference between these sequences as **shift**. A good time series data mining mechanism should be able to find similar sequences up to scaling and shifting.

Let *G* be a set of transformations then two sets of points are similar if there exists a transformation in *G* which maps one to the other. Let *D* be the distance metric between sequences and $\varepsilon > 0$ a tolerance. Query sequence *Q* is approximately similar within tolerance ε to data sequences when there exists a similarity transformation *T* so that $D(Q, T(S)) <= \varepsilon$, when ε is set to 0, we have exact similarity.

We consider an object to be a point in a multidimensional space. For non-point objects we assume there is a mapping function that maps every object to a point in the multi-dimensional space. Such a function is developed in many domains where multi-dimensional indexing has been used. For (e.g.,) Fourier Transform, Discrete Wavelet Transform for time series are some instances of the mapping function.

B. Similarity Transformations and Normal Forms

An n sequence *X* is a sequence (*x*1,….*xn*) of real numbers. A pair of reals (*a*, *b*) defines a similarity transformation $T_{a,b}$ over n-sequences by mapping each element x_i to $a \times (x_i + b)$. We will assume that all similarity transformations are non-degenerate; i.e, $a \neq 0$. In fact, we will further assume that $a > 0$, this restriction on a implies that a sequence symmetric to *X* w.r.t., *x*-axis is not considered similar to it. We say that *X* is similar to *Y* if there exists some

$(a, b)\ \varepsilon\ (R^+ \times R)$ such that $X = T_{a,b}(Y)$.

The similarity relation is reflexive, symmetric and transitive.

Reflexivity: for any sequence *X*,

$$X = T_{1,0}(X) \qquad [3]$$

[the identity transformation].

Symmetry: if $X = T_{a,b}(Y)$ then

$$Y = T_{1/a,\,-b/a}(X) = T^{-1}_{a,\,b}(X) \qquad [4]$$

[The inverse of $T_{a,b}$]

Transitivity: if $X = T_{a,b}(Y)$ and $Y = T_{c,d}(Z)$ then

$$X = T_{ac,\,ad+b}(Z) = (T_{a,\,b} \times T_{c,\,d}) \qquad [5]$$

[the non-commutative product of $T_{a,b}$ and $T_{c,d}$]

Therefore, the set of all sequences similar to a given one constitutes an equivalence class, we shall denote the similarity class of *X* by *X′*. The similarity relation partitions all n-sequences into similarity classes.

An *n*-sequence *x* is normal if $\alpha(x) = 0$ and

$\sigma(x) = 1$, where $\sigma(x)$ is the average of *x*

and $\sigma(x)$ is the standard deviation.

Let *x* be normal and *y* be similar to *x*. (i.e.) y = $T_{a,b}$ (*x*) for some $(a, b)\ \varepsilon\ (R^+ \times R)$,

then $\alpha(y) = b$ and $\sigma(y) = a$. *y* is normal only if $\sigma(y) = a = 1$ and $\alpha(y) = b = 0$; this is the identity transformation. This means that a similarity class has exactly one normal member, we will call it the normal form of all the members of the class.

Given any sequence *x*, $V(x)$ denotes the normal form of *x*. If $\alpha(x)$ is the average of *x* and $\sigma(x)$ is the standard deviation of *x*, then $x = \sigma(x) \times V(x) + \alpha$. Therefore we can compute $V(x)$ from *x* by the inverse transformation,

$$V(x) = T^{-1}_{o,\,a}(x)\ \ T_{1/a,\,a/a}(X) \qquad [6]$$

In a transformation $T_{a,b}$, we call a as the scale factor and *b* the shift factor. If *a* is 1, the transformation is a pure shift; if *b* is 0, it is a pure scaling. The identity transformation is a pure shift; the inverse of a shift is a shift; and the product of two shifts is also a shift. This allows us to conclude that the set of all shifts of a given sequence is an equivalence class. The same is true of the set of all scaling.

C. Semantics of Similarity Distance

Given two sequences X and Y, the similarity distance between X and Y is the distance between normal form of their respective similarity classes:

$$D_S\ (x, y) = D_E\ (V(x), V(y)) \qquad [7]$$

Where D_E is the Euclidean distance.

The similarity distance between any pair of sequences from x* and y* is the same; this gives us a distance metric for similarity classes:

$$D_S(x*, y*) = D_S\ (x, y) \qquad [8]$$

A distance metric should be non-negative and symmetric; and it should obey the triangle inequality. A good distance metric should also be effectively computable. It is easy to see that the similarity distance satisfies all these criteria. By using similarity distance the similarity semantics for the All-occurrences Subsequence Approximate Similarity is defined as follows:

Given a query sequence Q, a time series S, a tolerance $\varepsilon \geq 0$, and a similarity relation, find all contiguous subsequences S in the time series Š, such that $D_S\ (Q, S) \leq \varepsilon$.

Given a query sequence Q, a time series Š, and a similarity relation, find all contiguous subsequences S in the time series Š such that Q and S are similar. [belong to the same equivalence class]. The exact case can be answered using the normal forms, because Q and S are in the same equivalence class if and only if $V(Q) = V(S)$.

D. Constraint-Based Syntax of Similarity Queries

Constraint-based syntax for the general similarity query which expresses the queries is,

Given Q, ε, (l_a, U_a) and (l_b, U_b) find all (S, a, b) such that $D(Q, aS + b) \leq \varepsilon$,

where $l_a = a = U_a$ and $l_b = b = U_b$.

If the user want to query for scaling transformations only or for shift transformations only then :

Scaling: Find all $[S, a]$ such that $D(Q, aS) \leq \varepsilon$.

Shifting: Find all $[S, b]$ such that $D(Q, S + b) \leq \varepsilon$.

If the user want to use approximate matching queries or exact similarity queries then:

Approximate Match: Find all S such that

$D(Q, S) \leq \varepsilon$.

Exact Similarity: Find all $[S, a, b]$ such that

$Q = aS + b$.

V. EXPERIMENTAL STUDY

In this section, we present the results of an extensive empirical study that we have conducted using the adaptive framework and similarity search. The objective of this study is two fold:

1. to establish the effectiveness of our adaptive framework for data reduction in recent-biased time series
2. to demonstrate the benefits of using the similarity search methods for approximate and exact search queries.

A. Data Sets

Two data sets are used in our experiments. The first is time series "Teleccum" from MATLAB. It is real-world electrical consumption measured over the course of 5 days and the length of time series is 6000. The second is the time series of NASDAQ indices from Microsoft "MSFT". The daily close, open, high and low prices of indices are chosen and each time series is composed of 4,096 values from March 1986 to February 2006.

B. Evaluation Criterion

To evaluate the effectiveness of our method, a criterion is designed to measure the precision of approximation after data reduction. Assume that S and S' are the original and reconstructed time series. The error of approximation between S and S' is defined as the following:

$$E_{rr}\ (S' - S) = \mathrm{Dist}_{RB}\ (S' - S)\ /\ \mathrm{Dist}_{RB}\ (S - 0) \qquad [9]$$

Where Dist_{RB} is the recent-biased distance.

Steps Involved in Similarity Search

(i) Compress the stock data to get Open (*O*), High (*H*), Low (*L*) and Close (*C*) value for a given compression period.

(ii) Calculate the Level *L* of the DWT needed based on number of samples *N* in *C* of step *i*.

(iii) Perform a Level– *L DWT* on *C* based on results of step (i) and step (ii) to get D_i, $i = 1,2 \ldots L$

(iv) Scan the relation of wavelet coefficient sequentially to compare every sequence *S* to all the sequence that are after *S* in the relation. All the transformations are applied to every sequence during this comparison.

(v) Do the sequential scanning as instructed in the previous step, but stop the distance computation as soon as the distance exceeds ε.

C. Experimental Results

We have experimentally evaluated our method first on the Teleccum data streams. The effectiveness of our framework is shown by the experimental result shown by a time series of 6000 values. The time series is transformed with vari - DWT and then the time series is reconstructed from the reduced sequences.

Table 2. Number of coefficients for original data, reduced data and their difference using adaptive framework

	No. of Coefficients			
Time Series	1000	2000	4000	6000
Original data set	4.814	5.347	19.892	117.066
Reduced data set	4.746	4.94	12.157	60.956
Difference	–0.068	–0.407	–7.735	–56.11

Secondly we have experimentally evaluated our method for recent-biased analysis over online data streams. The effectiveness of our framework is shown by the experimental result shown by a time series of 512 values truncated from NASDAQ indices.

The time series is transformed with vari - DWT and then the time series is reconstructed from the reduced sequences. The original time series is partitioned into larger segments for older data, and the lengths of segments are 1,2,4,8,16,…. from recent to old data. At most four coefficients are kept for each segment in this method. For segments whose coefficients are less than four, all available coefficients are kept. The reconstructed time series after vari - DWT shows the effectiveness of capturing more details for recent data with smaller segments and long term changes at a coarser level. By keeping larger coefficients the parts of high energy are preserved.

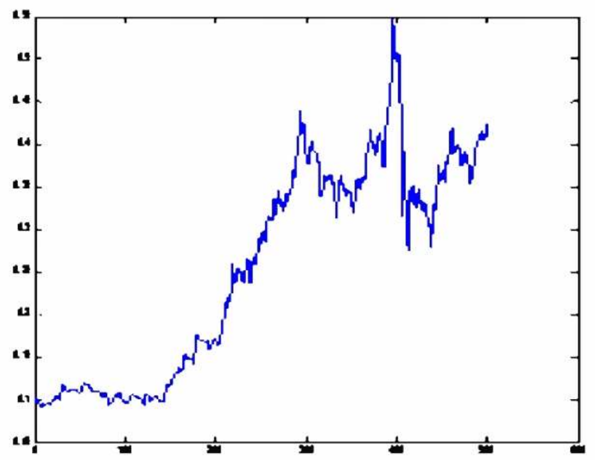

Fig. 2 Original time series after normalization

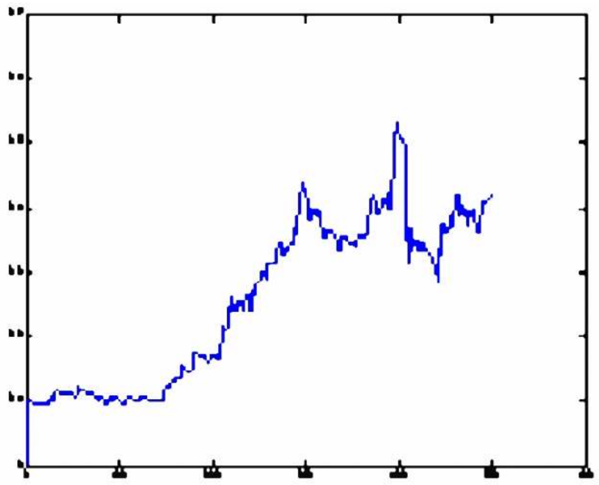

Fig. 3. Reconstructed time series

The above Fig. 2 and 3 show the Experimental results on a time series of Nasdaq indices. It is from June 1998 to November 1999, with a point per day and the length of the time series is 512. The vertical dotted lines are boundaries of segments.

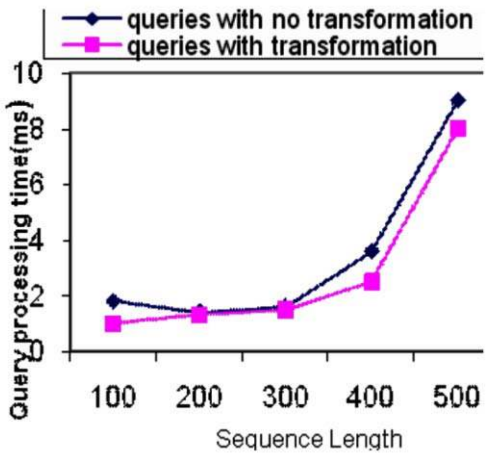

Fig. 4

Fig. 4 shows the comparison between the execution time for two kinds of queries such as range query with no transformation and range query with transformations by varying the length of sequences from 64 to 1024 while the no. of sequences are kept at 500.

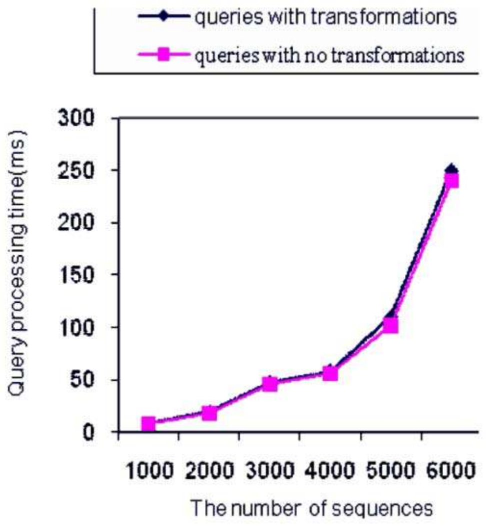

Fig. 5

Fig. 5 comparison between the execution time for two kinds of queries such as range query with no transformation by keeping the sequence length fixed to 128 while the number of sequences are varied from 1000 to 6000.

Fig. 6 comparison between the execution time for queries with transformation and sequential scanning with transformations by varying the length of sequences from 64 to 1024 while the no. of sequences are kept at 500.

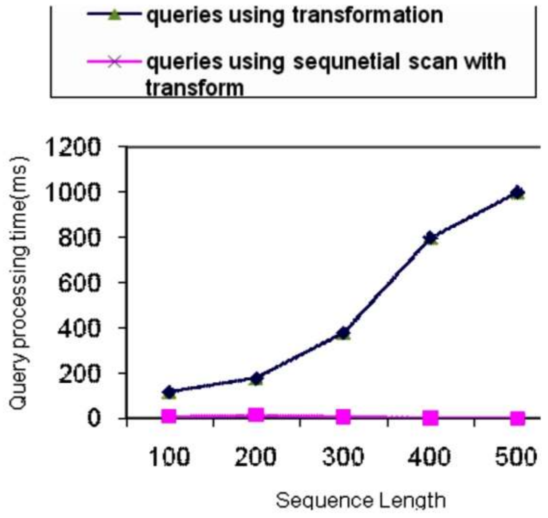

Fig. 6

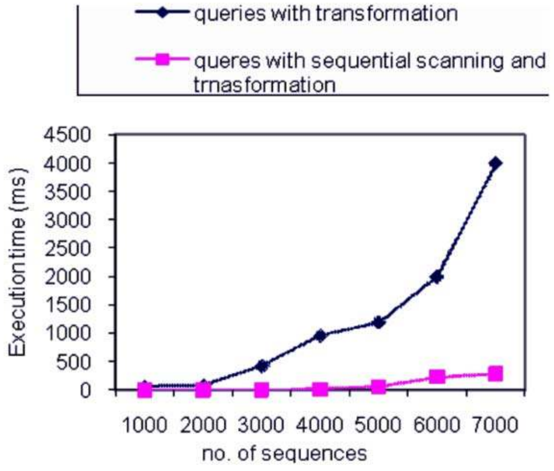

Fig. 7

Fig. 7 comparison between the execution time for two kinds of queries such as query with transformation and sequential scanning with transformation by keeping the sequence length fixed to 128 while the number of sequences are varied from 1000 to 7000.

VI. CONCLUSION

The method used for data reduction is Adaptive framework for online recent-biased time series analysis which uses vari-DWT, more details are kept for more recent data, while older data are kept at coarser level. With this method, the dimension of time series can be reduced and the efficiency can be greatly improved. Since only a small part of the time series needs to be processed when a data value arrives, high efficiency is achieved. In this approach, Wavelet-coefficient synopses are built and using these synopses we can provide approximate answers to queries. The Query processing method operates directly with the wavelet-coefficient, thus allowing fast processing of arbitrary complex queries entirely in the wavelet coefficient domain. A class of transformations have been used in a query language to express similarity search. This class allows the expression of several practically important notions of similarity and queries using this class can be efficiently implemented. One potential application which is emphasized in the paper is stock data analysis. The experiments show that the execution time of our method is almost same as that of accessing the index with no transformations; our method has much better performance than sequential scanning and the performance gets better by increasing both the number and length of sequences.

REFERENCES

[1] Agrawal, R. Faloutsos, C & Swami.A. 1993, Efficient similarity search in sequence databases, *Proceedings of the 4th Conference on Foundations Of Data organization and Algorithms.*

[2] B-K,Yi,H.V. Jagadish and C. Faloutsos, Efficient Retrieval of similar time sequences under time warping, in *Proc. Int. Conf. Data Engineering, Orlando, Florida, USA 1998, pp201-208.*

[3] K.P. Chan and A.W. Fu, 1999, Efficient Time Series Matching by Wavelets, *Proc. Int'l Conf on Data Eng. (ICDE 99).*

[4] L. Debnath and S. Nadarajah, Popular wavelet models, *Int. Journal of Wavelets &. Inf. Process, 4(4), 2006,pp 655 – 666.*

[5] Dina Q Goldin, C. Kanellakis, 1995, On Similarity Queries for Time-Series Data, *In Proceedings of ACM SIGMOD Int. Conf. Management of Data, pp 37-153.*

[6] Eamon Keogh, Kaushik Chakraborti, Michael Pazzani, Sharad Mehrotra, 2002, Locally Adaptive Dimensionality Reduction for indexing Large Time Series databases – *ACM Transactions on Database Systems.*

[7] Eamon Keogh, Kaushik.Chakrabti, Michael Pazzani and Sharad Mehrota, 2001, Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases, *Knowledge and Information Systems, vol 3, pp 263-286.*

[8] A. Guttman, R-Trees: A dynamic index structure for spatial searching, 1984, *In Proceedings of ACM SIGMOD Int. Conf. Management of Data, Boston, USA, pp 47-57.*

[9] F.Korn, H.V. Jagadish and C. Faloutsos, Efficiently supporting adhoc queries in large datasets of time sequences, in *Proc. ACM SIGMOD Int. Conf. Management of Data, Tucson, Arizona, USA , 1997, pp 289-300.*

[10] S. Kang, K.Jim, J.Chae, W.Choi, and S.Lee, Similarity search using the polar wavelet in time series databases, in *Proc. Int. Conf. Intelligent Computing, Qingdao, China, 2007, pp. 1347-1354.*

[11] T. Palpanas, M.Vlachos, E.Keogh, D.Gunopulos and W. Truppel, 2004, Online Amnesic Approximation of streaming Time Series*", Proc 20th Int'l Conf. on Data Eng.*

[12] D. Rafiei and A.O.Mendelzon, Similarity based queries for time series data, in *Proceedings of ACM SIGMOD Int. Conf.Management of Data, Arizona, USA 1997, pp 13-25.*

[13] D. Rafiei, On Similarity based queries for time series data, in *Proceedings of Int. Conf Data, Engineering, Sydney, Australia, 1997, pp. 410-417.*

[14] Yanchang Zhao, Shichao Zgang, 2006, Generalized Dimension Reduction Framework For Recent-Biased Time Series Analysis, *IEEE Transactions on Knowledge and Data Engineering, vol 18, No 2, February 2006.*

D. Muruga Radha Devi received the Bachelor's degree in Computer Science and Engineering from Bharathidasan University in 1994, Master's Degree in Computer Science and Engineering from University of Madras in 2001. Her research interests include time series analysis, efficient information retrieval and Data mining.