Knowl Inf Syst (2008) 15:131–147
DOI 10.1007/s10115-007-0097-z

REGULAR PAPER

A compact multi-resolution index for variable length queries in time series databases

Srividya Kadiyala · Nematollaah Shiri

Received: 10 January 2006 / Revised: 22 March 2007 / Accepted: 18 May 2007 /
Published online: 13 September 2007

Abstract We study the problem of searching similar patterns in time series data for variable length queries. Recently, a multi-resolution indexing technique (MRI) was proposed in (Kahveci and Singh, in proceedings of the international conference on data engineering, pp. 273–282, 2001; Kahveci and Singh, IEEE Trans Knowl Data Eng 16(4):418–433, 2004) to address this problem, which uses compression as an additional step to reduce the index size. In this paper, we propose an alternative technique, called compact MRI (CMRI), which uses adaptive piecewise constant approximation (APCA) representation as dimensionality reduction technique, and which occupies much less space without requiring compression. We implemented both MRI and CMRI, and conducted extensive experiments to evaluate and compare their performance on real stock data as well as synthetic. Our results indicate that CMRI provides a much better precision ranging from 0.75 to 0.89 on real data, and from 0.80 to 0.95 on synthetic data, while for MRI, these ranges are from 0.16 to 0.34, and from 0.03 to 0.65, respectively. Compared to sequential scan, we found that CMRI is 4–30 times faster and the number of disk I/Os it required is close to minimal. In terms of storage utilization, CMRI occupies 1% of the memory occupied by MRI. These results and analysis show CMRI to be an efficient and scalable indexing technique for large time series databases.

Keywords Time series · Multi-resolution index · Variable length queries · Similarity search · Query optimization · Performance · Dimensionality reduction · R-trees

1 Introduction

A time series is a large sequence of real numbers, in which each term represents a value, at a particular time, of some attribute such as price, temperature, etc. The popularity of time

S. Kadiyala · N. Shiri (✉)
Department of Computer Science and Software Engineering,
Concordia University, Montreal, QC, Canada
e-mail: shiri@cse.concordia.ca

S. Kadiyala
e-mail: kadiyalasvidya@yahoo.com; s_kadiya@cse.concordia.ca

series databases for predicting future events and trends is growing fast in many applications, including weather forecasting, stock market, genome, electrocardiograms, astronomical data, and video/audio data [21]. Such databases are normally very large since they store data frequently at regular times to preserve the history. As traditional sequential scan is inadequate for search in time series when the database is large, the demand to support efficient query processing over such databases is increasing. While much progress has been made, more work is required to develop more efficient and effective indexing structures and techniques. This has been our motivation in this work.

A good index should minimize the number of disk accesses (I/O operations), as this is the major factor that affects the performance of the index. An index on time series can be created by applying a dimensionality reduction technique on the data. For query processing upon the index, the search technique uses a distance measure to find the relative sequences to the query. To scale up to higher dimensions, a commonly used technique on time series is dimensionality reduction. The curse of dimensionality refers to the exponential growth of hyper volume as a function of dimensionality [4]. That is, reduction in the number of dimensions will increase the candidate set, which contains related sequences to a given query. The subset of this candidate set which only includes the query answers is called as the *result set*. The dimensionality curse of time series data makes it difficult to reduce the dimensions. Based on this several techniques are proposed for dimensionality reduction, including singular value decomposition (SVD) [7,13,16], discrete fourier transform (DFT) [1], discrete wavelet transform (DWT) [6,25], piecewise aggregate approximation (PAA) [16,26], adaptive piecewise constant approximation (APCA) [17], Multi-dimensional scaling [23], and Fast Map and its variants [8]. The particular application requirements and preferences will influence selection of an appropriate technique. Once a reduction technique is applied to data sequences, a multi-dimensional index structure is then created for the reduced space.

Similarity of time series can be measured using Euclidean or non-Euclidean metrics, or using the robust dynamic time warping (DTW) [14]. Euclidean methods define the distance between two points in a multi-dimensional space. In an N dimensional space, the Euclidean distance between points p and q is defined as $\sqrt{\sum_{i=1}^{N}(p_i - q_i)^2}$, where p_i (or q_i) is the coordinate of point p (or q) in dimension i [1,6,21]. Non-Euclidean metrics are distance measures that are not in a straight line but follow some rules; they can be used to measure the distance between more than two points. Non-Euclidean metrics, such as L_∞[2] and D_{norm}[18], are used for measuring the distance between two time series. Smoothing techniques are used to reduce irregularities, also called as random fluctuations, in time series data [21]. They provide a clear view of the true underlying behavior of the series. A common type of smoothing technique among others is moving average smoothing. Since seasonality may vary in general from series to series, so must the type of smoothing.

Time series databases use high dimensional index structures such as R-tree [10], R*-tree [3], KD-tree, VP-tree, and Grid files. These indexes in general partition either the data or the space into regions in such a way that makes it easier to direct the search to only regions which may contain good matches. Through dimensionality reduction, most of the information in the data is reduced to few dimensions, and only some of the principal components are used to build the index. There are two kinds of dimensionality reduction: global (GDR) and local (LDR). In a GDR technique, all the sequences in a database undergo the same dimensionality reduction. Such reduction techniques work well when the data is globally correlated. However, this may not always be the case, particularly when the regularity of data varies. In such cases LDR techniques are employed, which produce reduced sequences that carry more information [5,17] resulting in increased precision, where precision is defined as the ratio of sequences in the actual result to the candidate sequences obtained from the index. Chakrabarti and Mehrotra [5] introduced LDR through clustering, however it is proved later by Keogh and Lin [15] that clusterning is meaningless for time series. Later on they proposed a different technique for LDR, known as APCA [17], which improves the pruning power of an index using it.

There are two kinds of pattern matching queries often considered in time series data: range queries and nearest neighbor queries. In each case, we can have whole sequence matching and subsequence matching. There has been a growing demand in the aforementioned applications to support variable length queries for these kinds of pattern matching. It means the query length does not match with the length of sequence upon which the index is constructed and hence processing of the query is more time consuming. MRI is one of the best indexing technique proposed [11] which supports variable length queries for whole sequence matching. Whole sequence matching considers the entire query for processing by partition the query in different ways (depends on the partition algorithm). Each of these partitions are processed and the results from these are processed to give the final result set of the query.

In this paper, we propose an alternative index to support whole sequence matching, obtained by introducing two main changes to MRI, one to its complex grid structure and the other to the dimensionality reduction technique it uses. While maintaining the basic idea of MRI, we noted that many insertions of subsequences in MRI index are unnecessary and should be avoided. To fix this, we proposed a new design of the index structure which reduces the index size dramatically. As a by product of this change, our index is small and unlike in MRI, it does not impose the additional cost of compression. While MRI uses a GDR technique, we use APCA, which is a local dimensionality reduction technique. This resulted in improved precision, which translates to fewer disk I/Os.

The rest of this paper is organized as follows. Section 2 reviews existing indexing techniques for time series. Section 3 provides a background for our work, including the MRI index. In Section 4, we introduce our indexing technique. Section 5 presents the results of numerous experiments with real and synthetic data. Section 6 includes concluding remarks and suggestions for future work.

2 Related work

As we use dimensionality reduction as a step in our index construction, we first provide a brief description of these techniques.

Agarwal et al. [1] used DFT to transform time series into frequency domain. They consider only the first few coefficients to reduce the dimensionality and built an R-tree index. Chan and Fu [6] proposed Haar wavelet (DWT) transformation for dimensionality reduction and showed its advantages over DFT. Later on, Rafiei and Mendelzon showed that DFT can be improved by considering the symmetry of the Fourier transforms [22]. Wu et al. [25] showed that none of these transformations is superior but rather its performance depends on data.

Faloutsos et al. [9] introduced the I-adaptive index to address the matching problems in fixed length queries. By storing the sequences of fixed length in minimum bounding rectangles (MBR), they provided a solution for variable length queries longer than window size by using prefix search and multipiece search. While I-adaptive technique uses the prefix of the query to search the index where length of the prefix is equal to the window size, the multipiece search on the other hand splits the query sequence into non-overlapping subsequences of window size and then searches the index for all these subsequences. The search results are then combined to obtain the query result.

To improve the performance, Keogh et al. [16] introduced a new dimensionality reduction, called Piecewise Aggregate Approximation (PAA). They represent all the entries in a window by the average value of all the elements. Their experimental results showed that PAA can be computed faster than SVD, Haar, and DFT. The pruning power of PAA is similar to that of DFT and Haar, but poor compared to SVD. Popivanov and Miller [20] showed that PAA is poor compared to DFT, Haar wavelets, and Daubechies wavelets, and also that Daubechies family with different filters perform better than Haar wavelet. According to their results, Daubechies wavelets give the best precision and lowest number of disk accesses among the three techniques. Their results also indicate that performance of a dimensionality reduction technique used to create index depends on the type of data. Keogh et al. [17] proposed another dimensionality reduction technique, called APCA. When the data is not globally correlated, this technique improves the performance of the index by converting the sequence into several segments of different lengths. Li et al. [19] proposed Skyline index structure to improve the performance of dimensionality reduction techniques such as APCA by representing a group of related time series data according to their collective shapes, known as skyline bounding regions (SBR). They compute tight lower bounding distance based on SBR, and hence reduce the number of index pages accessed and the number of data objects fetched.

For variable length queries, Kahveci and Singh proposed a MRI and corresponding search methods to solve range and nearest neighbor queries [11]. This index is called multi-resolution because all the sequences in database are stored in the index at different resolutions. The index is a grid structure, in which each row stores the sequences at a particular resolution and each column stores a particular sequence at different resolutions. The index is created using different window sizes for every sequence by picking the length of the sequence just greater than the highest resolution in the index. The length of the sequence indexed at each row is 2^i for every integer i, $a \leq i \leq b$, where a and b are the lowest and highest resolution of the index.

The search algorithms proposed in [11] split the query into non-overlapping subqueries in such a way that the resolution of every subquery exists in the index. Range query algorithm searches the index for every resolution of the subquery and refines the threshold after each search. Their experiment results indicate that this method is 4–20 times faster than all other techniques. For K-Nearest Neighbor search algorithm, they first find the approximate distance for the Kth neighbor. Using this distance as threshold, they perform exact range search to find the K nearest neighbors. Since the size of their MRI index is large, the authors also proposed compression techniques.

Sun et al. [24] improved the compression of MRI through virtual bounding rectangle (VBR). VBRs are compact representation of the index nodes through quantization [24]. Compared to MBRs, their results show that VBRs can save 80–93% on the storage space for the index. They also improved the pruning power of the index by performing multipiece search in the descending order of the lengths of the query partitions.

3 Background

This section provides a technical background for our work. We provide a brief description of the APCA representation of time series, and then introduce the multi-resolution index structure (MRI), which we extended and improved in several ways. We will also recall the application of range and nearest neighbor queries on time series.

3.1 Adaptive piecewise constant approximation (APCA)

Global dimensionality reduction may not capture the entire information of a time series, in particular when the intensity of a segment in a time series changes over the time. The performance of an index can be improved if local dimensionality reduction is used instead. This is the key idea in the APCA reduction technique which allows representation of segments of different sizes with minimized approximation error [17]. To be precise, given a time series $s =< s_1, \ldots, s_n >$ of values, the APCA technique splits s into variable length segments, based on the data values, and represents s as a collection of segments each of which is a pair (m_k, i_k), where m_k is the mean of values in the kth segment and i_k is the index of the last value in that segment.

In our work, we use the APCA approximation with the distance measures proposed by Keogh et al. [17]. We use two distance measures they introduced, one to determine the distance D_{LB} between the query and a data item in a leaf node of the index, and the other measure to determine the distance $MinDist$ between the query and the MBR of the tree, which is an internal node. Interested readers can refer to [17] for details.

3.2 Multi-Resolution Index (MRI)

Multi-resolution index is a multi-dimensional index structure that supports range and nearest neighbor searching for variable length queries originally proposed in [11]. MRI uses multi-dimensional indexes at several resolutions, and is built as follows. For sequences $s_1, s_2, \ldots, s_n$ in the database, MRI stores a grid of I-adaptive trees $T_{i,j}$, where i ranges over resolutions a to b, and j ranges over the sequences s_1 to s_n. For this tree, MRI stores MBRs corresponding to database sequences at different resolution levels. More precisely, $T_{i,j}$ is the collection of MBRs for window size $w = 2^i$ (i.e., for resolution i) corresponding to sequence s_j. A separate tree $T_{i,j}$ is built for the MBRs of each sequence s_j at different resolution i. The ith row of the index is represented as the collection $R_i = T_{i,1}, \ldots, T_{i,n}$, and the jth column of the index is represented as the collection $C_j = T_{a,j}, \ldots, T_{b,j}$.

3.3 Range and nearest neighbor queries

Range query is the search for sequences that fall within a range/distance from the query sequence. For range query, a threshold value is provided to identify close enough sequences that match the query. In MRI, range search algorithm starts by partitioning the query sequence q of arbitrary length into a number of subqueries. A partial range query is then performed over all the partitions of the query at the corresponding resolution(2^i) in the multi-resolution index. The threshold value is updated for range search on the next partition after performing each partial range query on a partition. This process goes on until the last partition and the subsequences are retrieved based on the candidate set returned.

For a nearest neighbor query, MRI searches for k closest subsequences to the query sequence, where k is the number of desired neighbors. Nearest neighbor search is performed in two steps. First, it takes the longest prefix of q whose resolution appears in the MRI index and finds the k closest MBRs. The sequences are then read from these MBRs to find the actual distance, and the kth smallest distance (d) is considered the threshold value for the

Table 1 RMRI and MRI index structures at resolution 4

Sequence	RMRI	MRI
s_1	$s_{11}, s_{12}, s_{13}, \ldots, s_{1i}$	$s_{11}, s_{12}, s_{13}, \ldots, s_{1i}$
s_2	s_{2i}	$(s_{21}, s_{22}, s_{23}, \ldots), s_{2i}$
s_3	s_{3i}	$(s_{31}, s_{32}, s_{33}, \ldots), s_{3i}$

next step. In the second step, a range query is performed using the threshold value d. In our work, we use the same algorithm for nearest neighbor search.

4 Proposed indexing technique

A main concern with the MRI structure proposed in [11,12] is its size. This could be a problem when the time series data is very large and/or the index does not fit in the main memory. To address this, the authors of MRI used compression techniques to reduce the index size while preserving the information content. We take a different position and ask whether we could redesign the MRI structure so that its space utilization is reduced in such a way that while we continue enjoying capabilities of the index without requiring compression?

In our study, we found the answer to be positive. We noted that MRI inserts unnecessary subsequences into the index, as a result causes significant time and space overheads. We then propose a new multi-resolution index as an alternative to MRI which outperforms it in several ways, the index size, precision, and speed, without requiring compression. We refer to our technique as Compact MRI (CMRI, for short). Further, we use APCA as the dimension reduction technique as a step in our index construction technique which improves the precision of the index considerably. The CMRI structure stores at every resolution a tree, where these trees are constructed using the algorithm introduced in Sect. 4.2. To establish correctness of CMRI, we first propose to use an intermediate index structure, called Reduced MRI (RMRI), which uses a single tree at every resolution, as opposed to a collection of such trees. We describe RMRI in the following section.

4.1 Reduced multi-resolution index structure (RMRI)

Reduced multi-resolution index structure is a simplified index structure obtained from MRI by eliminating redundant sequences that are being inserted into MRI, explained as follows. Consider a time series database with sliding window of sequences $s_1, s_2, s_3, \ldots, s_n$ of size m. Suppose the minimum query length is 4, and a sequence length m is 68, where m lies between 2^6 and 2^7. Then the MRI index is created for resolutions 4, 8, 16, 32 and 64 with sliding windows of the same length.

The entries in the index at resolution 4 can be thought of as sequences of length 4 seen through a sliding window of size 4. Table 1 illustrates the sliding window of sequences in the data set inserted in the nodes of the MRI and RMRI structures at resolution 4. For clarity, a numerical example is also shown in Table 2, for a data set with a sequence of values from 1 to $1,000$.

Table 1 shows the initial three subsequences s_1, s_2, s_3 in the data set after they are inserted in the MRI and RMRI structures. The entries in each sequence are separated by commas. The table depicts the first three subsequences of the data set inserted into MRI and RMRI. For the RMRI, as can be seen, redundant subsequences are not present in the index

Table 2 Numerical example: RMRI and MRI structures at resolution 4

Sequence	RMRI	MRI
s_1: 1, 2, ..., 68	1–4, 2–5, ..., 65–68	1–4, 2–5, ..., 65–68
s_2: 2, 3, ..., 69	66–69	(2–5, 3–6, ...), 66–69
s_3: 3, 4, ..., 70	67–70	(3–6, 4–7, ...), 67–70

nodes. Table 2 shows the corresponding index nodes for the numerical example. The following explanation applies to both tables. Sequence s_1 in the database includes subsequences $\{s_{11}, s_{12}, s_{13}, \ldots, s_{1i}\}$ of length 4. Similarly s_2 includes $\{s_{21}, s_{22}, s_{23}, \ldots, s_{2i}\}$, where i is the number of sequences of length 4 in sequences of length m. Since the sequence is a sliding window, the first $i-1$ sequences $\{s_{21}, s_{22}, s_{23}, \ldots, s_{2i-1}\}$ of s_1 and the last $i-1$ sequences $\{s_{12}, s_{13}, s_{14}, \ldots, s_{1i}\}$ of s_2 are the same, and hence only s_{2i} is inserted in RMRI. Thus RMRI does not store unnecessary nodes which MRI inserts, indicated in parentheses in the MRI column. We deduce that while RMRI and MRI include the same pointers to sequences, the size of RMRI is much less than MRI. This advantage of the RMRI index structure for its reduced storage requirement allows RMRI to handle larger time series and results in better search performance, shown also by our experimental results discussed later.

In other words after eliminating the insertion of these duplicate sequences in RMRI, only the first column have the I-adaptive index while the rest of the columns have only one leaf node. Besides, RMRI uses a local dimensionality reduction technique, as opposed to a global dimensionality reduction technique employed in MRI. This improves the recall and precision rates of RMRI over MRI, which is explained later.

Construction of RMRI proceeds as follows. First we use APCA as the local dimensionality reduction technique, which maps a small subset of points in an M dimensional space to points in $2M$ dimensional space. As in MRI, our index created for resolutions a to b, is described as follows. Let s be the longest sequence in the database, where $2^b \leq |s| \leq 2^{b+1}$. We use the minimum possible length of a query to be 2^a, for some integer a, where $a \leq b$. We apply Haar wavelet transformation (HWT) for every sequence s in the database. The result of this transformation of sequence s is used to create the APCA representation of s, as suggested in [17], to find variable length segments. For each sequence s in each data set, an I-adaptive index is created for all the subsequences of s except the redundant subsequences from resolutions a to b. The resulting index is RMRI, whose structure is shown in Fig. 1. The correctness of the RMRI index follows from the correctness of the APCA representation, which is an established dimensionality reduction technique.

4.2 Compact multi-resolution index structure (CMRI)

In addition to using APCA representation in RMRI, we investigated other possible ways to further improve the index, and reduce its size. From Tables 1 and 2, we noted that the structure of the index nodes at every resolution for MRI could be further reduced to just one node for all data sequences but the first, without loss of information. That is, the leaves of the I-adaptive index in RMRI will have the same information as the leaves in MRI. This ensures that there will be no false dismissals or false hits in using RMRI. We further combined all the nodes at each resolution in RMRI to form an I-adaptive index, as shown for resolution 2^a in Fig. 2. In other words, we have a single I-adaptive index at each resolution for the entire time series data. The structure of the resulting index structure is called *Compact MRI* (CMRI, for short) and is shown in Fig. 2. We emphasize that the purpose of the introduction of RMRI

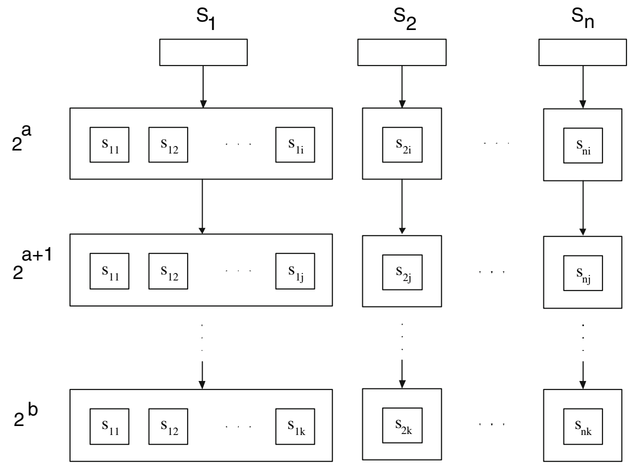
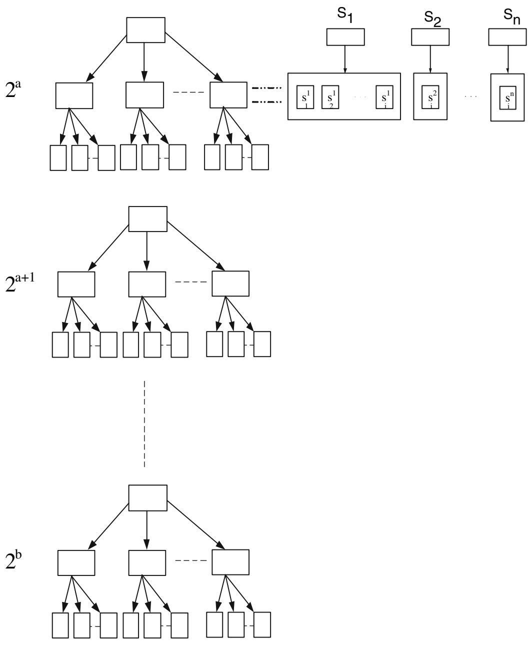


Fig. 1 Structure of reduced MRI (RMRI)

Fig. 2 Structure of compact MRI (CMRI)



was conceptual; to serve as an intermediate structure relating MRI and CMRI in order to establish correctness of CMRI w.r.t MRI. In what follows, we propose an algorithm which directly constructs the CMRI index from the input time series, by inserting the tree nodes into an I-adaptive structure at various resolutions. This is done in $O(n)$ for the simple structure of CMRI as opposed to the complex grid structure of MRI, where n is the number of sequences to be indexed. This construction is formally described as the following algorithm, where the database is the collection of the input time series data.

```
Algorithm Create CMRI($s$):
Input: Database, resolution range $a$ to $b$, and dimension $M$;
Output: M dimensional index structure for resolutions $a$ to $b$;

$S_1$. For every sequence $s$ in the database:
  $S_{1.1}$. For every resolution $a$ to $b$ of sequence $s$:
    $S_{1.1.1}$. Apply the APCA transformation on $s$;
    $S_{1.1.2}$. Insert $s$ into $2M$ dimensional index tree at corresponding resolution with pointer
             to location of $s$ in the database.
End Create;
```

4.3 Range queries

We propose a range search algorithm for CMRI by adopting the corresponding algorithm proposed for MRI in [12]. Instead of traversing the trees for each sequence as in MRI, CMRI traverses the I-adaptive index at a particular resolution to find similar patterns with all sequences in the database. An important consequence of the simple structure of CMRI is that its range search algorithm has just a single loop whereas it is a double loop in MRI. The search algorithm in MRI takes more time as the search proceeds to next sequence after the current one is complete, whereas in CMRI, a search on the tree at corresponding resolution is performed. the index nodes in the trees at every resolution are combined into one. That is, there will be less chance in MRI to prune away a group of nodes at once. In our case, we do not have to examine the data sequence by sequence in order to find similar patterns but rather need to traverse tree by tree based on the resolution. The above steps are formalized into the following range search algorithm for CMRI.

```
Algorithm RangeQuery CMRI ($q, \epsilon$):
Input: Query $q$ and threshold $\epsilon$;
Output: Result set which includes the time sequences which match $q$.

$S_1$. Partition query $q$ of length $|q|$, where $|q| \geq 2^a$, into $p$ parts as $q_1, q_2, \ldots, q_p$ in ascend-
     ing order of their length such that $|q_j| = 2^k$, for $a \leq k \leq b$ and $1 \leq i \leq p$;

$S_2$. Initialize $\epsilon_p = \epsilon$, where $\epsilon$ is threshold for range queries;

$S_3$. For $i := 1$ to $p$:
  $S_{3.1}$. Perform query $q_i$ on tree $T_k$ at resolution $k$, where $k = \log_2 |q_i|$. Let $\mathrm{Res}_i$ be the
        result set for partition $i$;
  $S_{3.2}$. Set $\epsilon_{i+1} := \max_{B \in \mathrm{Res}_i} \{\sqrt{\epsilon_i^2 - d(q_i, B)^2}\}$, where $d(q_i, B)$ is the distance between
        query subsequence $q_i$ and bounding region B;

$S_4$. Filter the results of each partition to find the match for $q$ of length $|q|$;

$S_5$. Read disk pages indicated in $\mathrm{Res}_p$ and perform post-processing to eliminated false ret-
     rievals.
End RangeQuery;
```

The algorithm for k-nearest neighbor search is adopted from MRI; interested readers can refer to [12] for more details. Due to these improvements the precision of resulting CMRI is identical to that of RMRI, since the information prevailed in the I-adaptive index is the same as that stored in RMRI. This is also observed in our extensive experiments. On the other hand, the speed of CMRI is significantly improved since a group of nodes can be eliminated (or considered) from the search. The space utilization by RMRI and CMRI is almost the same, and both use much smaller space compared to MRI for the same time series database.

5 Experiments and results

To evaluate the merits of CMRI indexing technique, we have implemented in C++ both CMRI and the MRI index as proposed in [12]. For these experiments, we used a PC with a Pentium 4 processor at 2.80 GHz, 512 MB RAM, 140 GB disk space, on Windows XP. The buffer size considered for disk I/Os was 8 KB. A measure of disk I/Os is also provided as it is customary to report time in terms of disk I/O's [6,21]. We performed extensive experiments on real and synthetic time series in order to evaluate and compare the various aspects of these two indexing schemes including precision, efficiency, memory utilization, and scalability. We compare CMRI with the basic MRI technique and also compare them using the search algorithm proposed in COMRi [24] that looks for the matching sequences in the descending order of the lengths of the partitions of the query. We refer to this algorithm as "descending search". This section presents the experiments and the results.

For the real data, we collected stock market information from the Yahoo web site at http://finance.yahoo.com. This data includes about 200,000 sequences in 201 collections/stocks with total size of 1.53 MB. For the synthetic data, we generated 200,000 sequences in 271 collections with total size of 2.60 MB. This was done by generating, for each collection, a random seed $x = random(200)$, which was then altered by introducing some noise to generate other values $x' = x + random(5)$ in the collection.

As in the context of MRI, we also studied effectiveness of the index when the data size increases using AUSLAN data from http://kdd.ics.uci.edu/. This data set consists of sample Australian sign language with 2,565 signs of various lengths, out of which 2,090 signs of size slightly more than 20 MB are considered for the experiments.

Several experiments are performed to study the effects of MRI and CMRI for range queries on large data sets. Queries of variable length ranging from 16 to 1,024 are considered for the experiments. We have considered range and nearest neighbor queries of variable lengths in our experiments, details of which are provided in the following sections.

We consider the prefix search and compare the results of our range query experiments, which show that prefix search is not better than multipiece search.

5.1 Range queries

For range queries, we considered resolutions of the index ranging from 4 to 8 i.e, {4,5,6,7,8}. The queries considered were also of varying lengths ranging from 16 elements/values to 1,024. Each query sequence was generated by randomly selecting a sequence of random length from the database and modifying it by about 10%. For each query sequence, we randomly produced the radii in the range [0.1,1]. We assume that the lengths of queries are evenly distributed between 16 and 1,024. As for the number of dimensions chosen from the sequence to build the index, we considered four different dimensions: 4, 6, 8, and 10.

We remind the reader that "dimension" is the value M chosen after the Haar transformation, and not the number of dimensions used in the indexing of the APCA representation, which is $2M$. For a query over a data set, the *candidate set is defined as the set of sequences similar to query returned by the index* and *result set is defined by the actual number of*

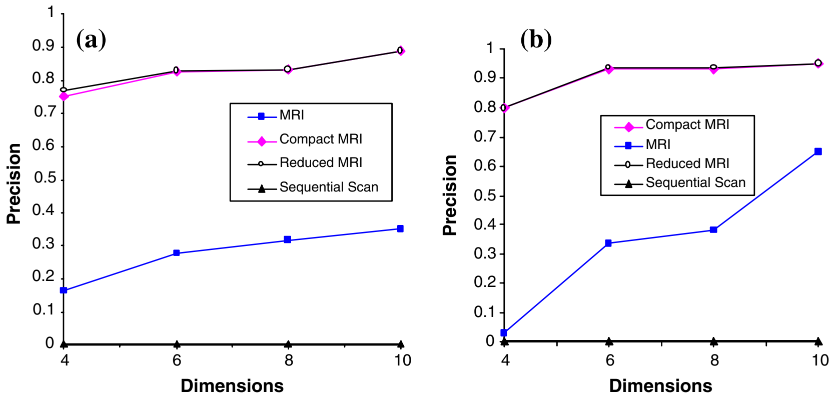


Fig. 3 Precision for range queries on **a** real data and **b** synthetic data

sequences that match the query. Using this, the notion of **precision** *is defined as the ratio of the number of sequences in the result set to the number of sequences in the candidate set*. As in [12], the precision and number of disk I/O measures reported in our work are the average of the corresponding measures taken over 100 queries. The aforementioned parameters, assumptions, and measures are valid in all our experiments on MRI, RMRI, and CMRI. The assumptions and parameters are not required for sequential scan—a well-known query processing technique. The experimental results show that precision and the number of disk I/Os for RMRI and CMRI are both far better than MRI.

The graph in Fig. 3 shows precision for the four techniques, RMRI, CMRI, MRI, and sequential scan at different dimensions on both real data (a) and synthetic data (b). Since RMRI and CMRI have the same precision, their graphs in the figure coincide, even the disk I/O's are same. Because of this, in the rest of this paper, we will only consider CMRI in our experimental results and comparisons. As shown in Fig. 3, while the precision of MRI ranges from 0.16 to 0.34 for real data and 0.03–0.65 for synthetic data, the precision of CMRI is far better, ranging from 0.75 to 0.89 for real data and 0.80–0.95 for synthetic data. Sequential scan has the least precision and most number of disk I/Os as it scans the entire database.

The reason for improvement in the precision with CMRI is a result of using APCA representation. When using APCA dimensionality reduction technique, the index preserves more information which helps in narrowing the search, thus increasing the precision and decreasing the disk I/Os.

The same set of experiments are performed on MRI and CMRI with prefix search and descending search and the results are shown in Fig. 4. The results for these experiments are shown in different graphs to have clear understanding of their behavior. In the figure prefix search over MRI and CMRI is represented by PrefixMRI and PrefixCMRI. In prefix search, as the name indicates only the prefix of the query whose length is equal to the (maximum available: 2^b) resolution of the index is considered for the search, while the rest of the portion of the query is not used untill the sequences are retrieved from the database to find the exact distance. Prefix search on MRI yields poor precision due to the use of DWT on MRI, almost close to zero, where as on CMRI precision of prefix search is far greater than precision of multipiece search on MRI. Since CMRI preserves more information because of the use of APCA, precision of prefix search on CMRI is close to precision of multipiece search on CMRI. Descending search algorithm, which treats multipiece search in descending order

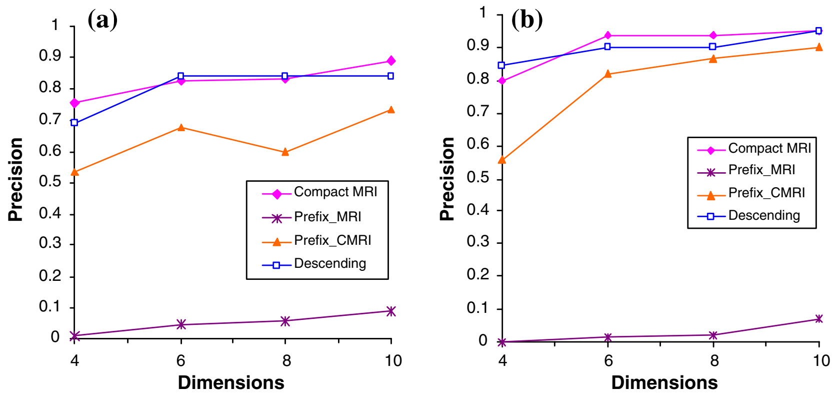


Fig. 4 Precision for range queries on **a** real data and **b** synthetic data

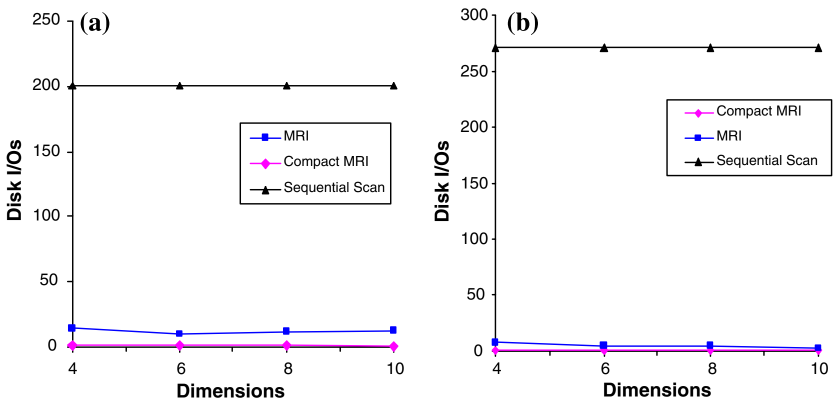


Fig. 5 Disk I/Os for range queries on **a** real data and **b** synthetic data

of the length of subqueries, has almost the same performance as MRI's search algorithm, multipiece search is performed in ascending order of the length of subqueries, on CMRI.

Figures 5 and 6 show the number of disk I/Os performed by multipiece search and prefix search on CMRI, MRI, descending search, and sequential scan, at different dimensions for the same range queries. As we see, CMRI outperforms MRI by performing fewer disk accesses both on real data (a) and synthetic data (b). The graphs indicate that CMRI has noticeably greater pruning power than MRI. In case of real data, the number of disk I/Os for CMRI is almost identical to the number of sequences in the query result. For synthetic data, while the number of disk I/Os for MRI decreases with increase in the number of dimensions, CMRI consistently requires very few disk I/Os at every dimension. Since sequential scan reads the entire database, the number of disk I/Os remains high and equals the number of pages read for the entire database. These figures clearly show that multipiece search has higher precision than prefix search. The number of disk I/O's for descending search, prefix search on CMRI, and multipiece search on MRI are similar as their graphs almost coincide. We thus exclude prefix search and descending search in our experiment results and comparisons below.

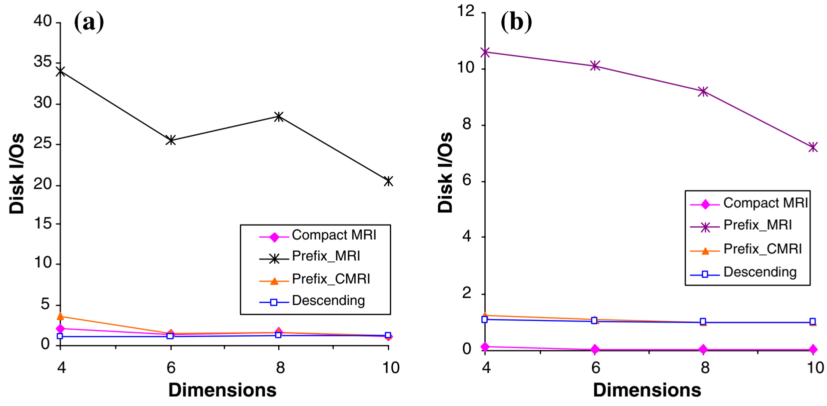


Fig. 6 Disk I/Os for range queries on **a** real data and **b** synthetic data

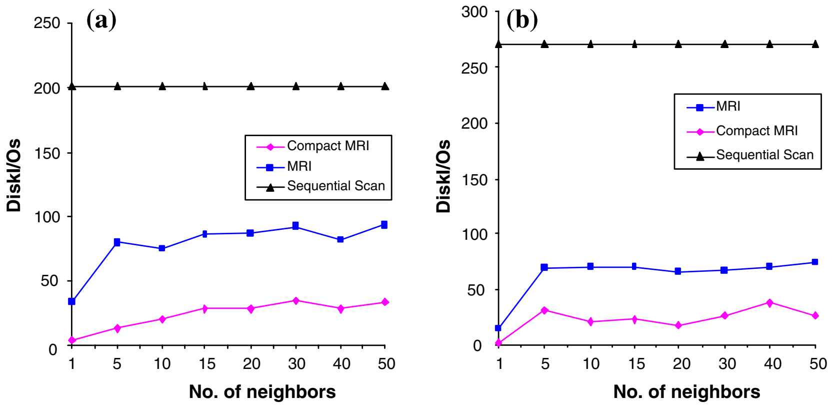


Fig. 7 Disk I/Os for nearest neighbor queries on **a** real data and **b** synthetic data

5.2 Nearest neighbors queries

Nearest neighbor queries are performed by varying the number neighbors. Even though both CMRI and MRI have highest precision at dimension 10 we consider dimension 8 as optimal for search because the increase in performance from dimensions 8–10 is not much, considering the effect of memory utilized and speed of search. Hence we fixed the number of Haar coefficients to be 8, that is, the number of dimensions considered for creating the index is 8. Experiments are performed on the indexes MRI and CMRI at dimensionality 8 for number of neighbors $K = \{1, 5, 10, 15, 20, 30, 40, 50\}$ on real and synthetic data. We also considered sequential scan for the comparison. Number of disk I/Os consumed by these techniques is used as a comparison metric. Figures 7 shows the number of disk I/Os for CMRI, MRI, and Sequential scan. While the number of disk I/Os in MRI ranges from 33 to 92 for real data (a), it ranges from 4 to 35 for CMRI. That is, the maximum disk I/Os consumed by CMRI is close to the minimum disk I/Os consumed by MRI. Also for the synthetic data shown in (b), CMRI outperforms MRI significantly.On the synthetic data, the number of disk I/Os for MRI ranges from 14 to 75, while it ranges from 2 to 26 for CMRI. The reason for better results of CMRI is its ability to prune unnecessary sequences with the use of information prevailed in

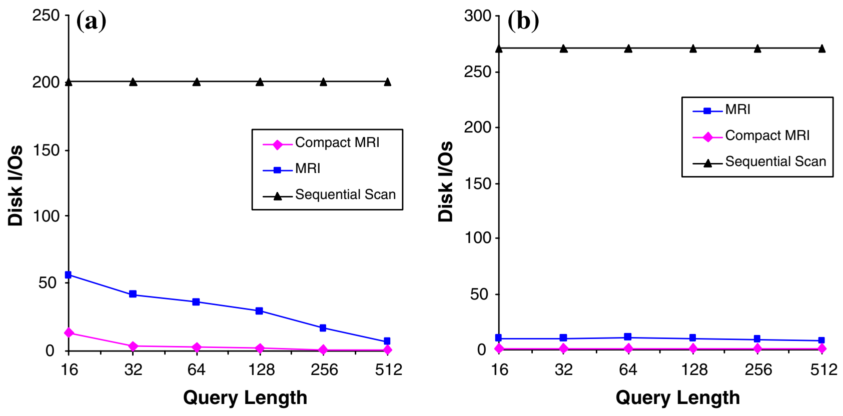


Fig. 8 Disk I/Os for range queries at different lengths on **a** real data and **b** synthetic data

the index, thus yeilding less number of false hits and reduces the number of disk I/Os to be performed, when compared to that of MRI. For the sequential scan, the number of disk I/Os required is same as the number of blocks occupied by real and synthetic data.

5.3 Performance on variable length queries

We have also evaluated effectiveness of CMRI, MRI, and sequential scan for variable length queries on real and synthetic data. For this set of experiments, we considered queries of length in $L = \{16, 32, 64, 128, 256, 512\}$, and for MRI and CMRI, we constructed the index at dimension 8, and generated random queries and threshold values for each query length, as explained for range queries. We messured the number of disk I/Os by averaging 100 queries for indexing different techniques. Figure 8 illustrates the experiment results. The number of disk I/Os performed by these techniques on real data is shown in (a). As can be seen, CMRI outperforms MRI with at least one order of magnitude, and outperforms sequential scan with two order of magnitudes. This indicates that the pruning power of CMRI is significant. Figure 8b shows the experiment results for synthetic data. Here again, CMRI has the least number of disk I/Os. The reason for it being CMRI results in fewer number of false hits when compared to the MRI index structure. Since the false hits are less so are the number of disk I/O's. CMRI is an order of magnitude better than MRI on both real and synthetic data. Moreover, as observed in the above experiments, the performance of CMRI is consistent and similar on both real and synthetic time series data.

5.4 Performance on large time series data

We have also compared efficiency of CMRI, MRI and sequential scan on large data sets. We carried out a c number of experiments for variable length queries of length {16, 32, 64, 128, 256, 512} on AUSLAN data set of size 20 MB, and measured the number disk I/Os and also recorded the real time it took to process the search queries. For these experiments we generated for each query the radii in the range (0.1, 0.2). The results are taken as average of 100 such queries and are shown in Fig. 9. In this figure, (a) shows the number of disk I/O's performed by CMRI and MRI. Clearly CMRI performed significantly much fewer disk I/Os compared to the other two techniques; an order of magnitude better than MRI. Figure 9b shows the search time by MRI, CMRI and sequential scan in seconds.

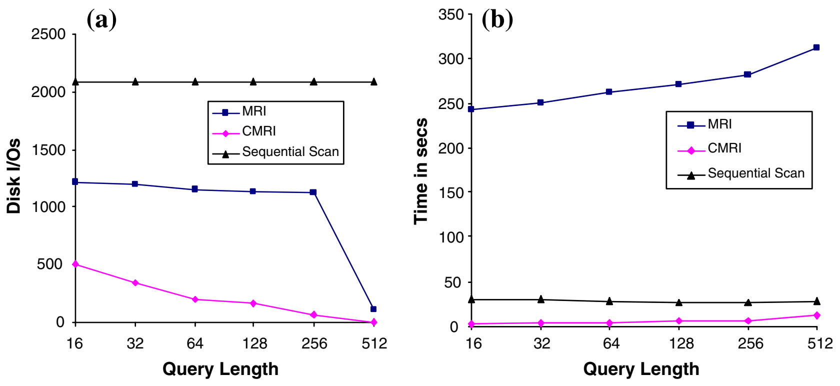


Fig. 9 Disk I/Os **a** and time (in seconds) **b** for range queries of variable lengths on AUSLAN data

From the figures notice that the disk I/O decreases significantly when query length increases while the wall clock time increases. In MRI and CMRI, every index node has lower and higher boundary with each of them having *X* co-ordinate points. For any query length the index has to decide either to consider a particular node for further processing or not with these two boundaries. When processing of nodes is compared with query length, less number of nodes are processed for shorter query length. As the length of the query increases, fluctuations of the sequences are less captured by these 2 boundaries, the number of nodes to be processed during the search increases along with the length resulting in the increase in the processing time. Disk I/Os decreases with the query length because the number of matches for the query decreases with the increase in the query length.

The figures show clear superiority of CMRI over MRI and sequential scan with disk I/Os and speed. This is due to the fact that MRI has to process more nodes including the redundant nodes stored in its complex index structure as the nodes for each resolution are spread over many I-adaptive indices whereas for CMRI, all the nodes at each resolution are grouped into a single I-adaptive index, resulting in increased efficiency.

As our results indicate, CMRI has better precision with fewer number of disk I/Os. This improvement is due to APCA, a local dimensionality reduction technique we used in CMRI. To verify this, we used the DWT technique, instead of APCA, in conjunction with our reduced index structure, and carried out several experiments. Our results indicated that even though APCA needs more computation time than DWT, it increases the precision of the index without any adverse affect on the search performance. There is adverse impact because APCA narrows the search and hence accesses fewer nodes in the same time while it performs more computation in the time saved. While with DWT the search accesses more nodes, causing more disk I/Os and hence increased time, as I/O operations are costly.

All these experiments show that a simple mutli resolution index structure CMRI with a proper dimensionality reduction technique can outperform a complex grid structure of MRI.

6 Conclusion and future work

In this paper we introduced an indexing technique called CMRI, to support variable length queries for time series data. To obtain CMRI, we first introduced an intermediate structure, RMRI, by reducing the contents of the MRI nodes and then combined the reduced and separate nodes related to each sequence in the database into a R-tree. Performing numerous experiments on real-life and synthetic data, we showed that CMRI with a simple R-tree at every resolution outperforms MRI, the best known indexing technique which supports variable length queries on time series. Unlike MRI which keeps track of the number of sequences or number of trees in its index and uses this information during a search, CMRI does not record or use this information due to its simple structure. An immediate consequence of this is that the size of CMRI is much reduced and hence does not require any compression. Note that this is true despite the fact that with APCA we use $2M$ dimensions for every M coefficients considered. However the produced index is 1% of the size of the corresponding MRI, which is a significant reduction. In order to apply the DWT and APCA transformation, the data should first be converted to real numbers. The size of CMRI index is therefore proportional to size of data irrespective of the characteristics of the data—an indication of robustness of CMRI. The small size of CMRI results in increased efficiency, reduced disk I/Os, and improved precision, as indicated by our experiment results.

As a future work, we would like to use CMRI for times series data in other applications mentioned in the introduction. In particular, we are interested in using CMRI for searching DNA and protein sequences to study the performance of CMRI compared to suffix tree and suffix array solutions, which are well-known indexing techniques for sequence data in bioinformatics research.

Acknowledgements This work was supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada and by ENCS faculty, Concordia University. We also thank anonymous reviewers for their useful comments.

References

1. Agarwal R, Faloutsos C, Swami A (1993) Efficient similarity search in sequence databases. In: Proceedings of the International conference on foundation of data organization and algorithms, pp 69–84
2. Agrawal R, Lin KI, Sawhney HS, Shim K (1995) Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In: Proceedings of the 21st international conference on very large data bases. Morgan Kaufmann, Zurich, pp 490–501
3. Beckman N, Kriegel HP, Schneider R, Fu W (1990) The R*-tree: an efficient and robust access method for points and retangles. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 322–332
4. Bellman RE (1961) Adaptive control process: a guided tour. Princeton University Press, Princeton
5. Chakrabarti K, Mehrotra S (2000) Local dimensionality reduction: a new approach to indexing high dimensional spaces. In: Proceedings of the 26th VLDB Conference
6. Chan K, Fu W (1999) Efficient time series matching by wavelets. In: Proceedings of the 15th International conference on data engineering, pp 126–133
7. Faloutsos C (1996) Searching multimedia databases by content. Kluwer, Dordrecht
8. Faloutsos C, King-Ip (David) Lin (1995) Fastmap: a fast algorithm for indexing, data mining and visualization of traditional and multimedia datasets. In: Proceedings of the ACM SIGMOD conference, pp 163–174
9. Faloutsos C, Ranganathan M, Manopolous Y (1994) Fast subsequence matching in time series databases. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 419–429
10. Guttman A (1984) R-trees: An efficient indexing structure for spatial searching. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 47–57
11. Kahveci T, Singh AK (2001) Variable length queries for time series data. In: Proceedings of the international conference on data engineering, pp 273–282
12. Kahveci T, Singh AK (2004) Optimizing similarity search for arbitrary length time series queries. IEEE Trans Knowl Data Eng 16(4):418–433

13. Kanth KVR, Agrawal D, Singh A (1998) Dimensionality reduction for similarity searching in dynamic databases. In: Proceedings of the ACM SIGMOD international conference on management of data
14. Keogh E, Ann RC (2005) Exact indexing of dynamic time warping. Knowl Info Syst 7(3):358–386
15. Keogh EJ, Lin J (2005) Clustering of time-series subsequences is meaningless: implications for previous and future research. Knowl Info Syst 8(2):154–177
16. Keogh E, Chakrabarti K, Mehrotra S, Pazzani M (2001) Dimensionality reduction for fast similarity search in large databases. Knowl Info Syst 3(3):263–286
17. Keogh E, Chakrabarti K, Mehrotra S, Pazzani M (2001b) Locally adaptive dimensionallity reduction for indexing large time series databases. J ACM
18. Lee S, Chun S, Kim D, Lee J, Chung C (2000) Similarity search for multidimensional data sequences. In: Proceedings of the 16th international conference on data engineering. IEEE Computer Society, Washington, DC, pp 599–608
19. Li Q, Lopez IFV, Moon B (2000) Skyline index for time series data. IEEE Trans Knowl Data Eng (TKDE) 16:669–664
20. Popivanov I, Miller RJ (2002) Similarity search over time series database using wavelets. In: Proceedings of the 18th international conference on data engineering, pp 212–221
21. Rafiei D, Mendelzon AO (1997) Similarity-based queries for time series data. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 13–25
22. Rafiei D, Mendelzon AO (1998) Efficient retrieval of similar time sequences. In: Proceedings of the FODO Conference, pp 249–256
23. Shepard RN (1980) Multi dimensional scaling. Princeton University Press, Princeton
24. Sun H, Ozturk O, Ferhatosmanoglu H (2003) Comri: a compressed multi-resolution index structure for sequence similarity queries. In: IEEE computer society bioinformatics conference (CSB'03), pp 553–558
25. Wu YL, Divyakant A, Abbadi AE (2000) A comparision of dft and dwt based similarity search in time series databases. In: Proceedings of the international conference on information and knowledge management, pp 488–495
26. Yi BK, Faloutsos C (2000) Fast time sequence indexing for arbitrary lp norms. In: Proceedings of the 26th international conference on very large databases(VLDB)

Authors Biography

Srividya Kadiyala is currently a Software Developer at SAP in Montreal, Canada. She received the BSc Degree in Computer Science from Nagarjuna University, Vijayawada, India, in 2002, and the MSc degree in Computer Science from Concordia University in 2006.

Nematollaah Shiri received the BSc degree in Computer Science from Sharif University of Technology (1985), Iran, the MSc in Computer Science from McGill University (1992), Canada, and the PhD in Computer Science from Concordia University (1997), Canada. He is currently an associate professor in the Computer Science and Software Engineering Dept. at Concordia University. His research interests include databases and knowledge base systems, uncertainty, query processing and optimization, bioinformatics, data and web usage mining. He is a member of the ACM and the IEEE Computer Society.