Skyline Index for Time Series Data

Quanzhong Li, Inés Fernando Vega López, and Bongki Moon

Abstract—We have developed a new indexing strategy that helps overcome the curse of dimensionality for time series data. Our proposed approach, called *Skyline Index*, adopts new *Skyline Bounding Regions (SBR)* to approximate and represent a group of time series data according to their collective shape. Skyline bounding regions allow us to define a distance function that tightly lower bounds the distance between a query and a group of time series data. In an extensive performance study, we investigate the impact of different distance functions by various dimensionality reduction and indexing techniques on the performance of similarity search, including index pages accessed, data objects fetched, and overall query processing time. In addition, we show that, for k-nearest neighbor queries, the proposed Skyline index approach can be coupled with the state of the art dimensionality reduction techniques such as Adaptive Piecewise Constant Approximation (APCA) and improve its performance by up to a factor of 3.

Index Terms—Data approximation, dimensionality reduction, similarity search, skyline bounding region, skyline index, time series data.

1 Introduction

A time series data is a (potentially long) sequence of real values, each of which represents a value measured at a point in time. Due to the time-varying nature of the universe, there are countless examples of time series data from diverse sources and applications, such as stock prices, currency exchange rates, electrocardiograms, and gene expression measurements. With the growing popularity of time series data, there is an increasing demand to support fast retrieval of time series data based on similarity measurements. For example, a fund manager of a stock brokerage firm may be interested in finding all stocks whose prices moved similarly to that of a particular stock or following a certain pattern (e.g., head-and-shoulder). Therefore, the problem of indexing and searching time series data has been the focus of many research activities in the database community for the past few years.

Similarity search, or query by content, for time series data can be classified into *whole sequence matching* and *subsequence matching*. In whole sequence matching, the time series to be compared have the same length, whereas, in subsequence matching, we look for a consecutive subsequence within the data that best matches the query sequence. While different measures have been proposed to estimate similarity between two time series [2], [9], [28], the Euclidean distance and, in general, L_p norms have received most of the attention. However, when the sizes of two time series are different or when it is required to match sequences that are locally out of phase, using L_p norms may not produce the desired results. In such cases, Dynamic Time Warping (DTW), a more robust distance measure, is used instead [4], [16], [20], [22], [27], [37].

- *Q. Li and B. Moon are with the Department of Computer Science, University of Arizona, Tucson, AZ 85721. E-mail: {lqz, bkmoon}@cs.arizona.edu.*
- *I.F.V. López is with the Escuela de Informática, Universidad Autonóma de Sinaloa, Culiacán, Sinaloa, México. E-mail: ifvega@uas.uasnet.mx.*

Manuscript received 28 Aug. 2003; revised 6 Dec. 2003; accepted 16 Dec. 2003.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0162-0803.

For similarity search, the user specifies the number of entries to be included in the answer. The search algorithm will retrieve from the database k entries that are the most similar to the query time series q. This type of similarity query is known as k-nearest neighbor (k-NN) search. Alternatively, the user could specify a tolerance ϵ, or degree of similarity, for the entries in the answer. In this case, the search algorithm will retrieve all entries in the database within a distance ϵ from q. This type of query is known as ϵ-range query. In any case, the solution to the search includes all entries in the database that satisfy the following condition:

$$Dist(x, q) \leq r,$$

where $Dist()$ is a domain-specific distance function used to estimate the similarity between two time series, x is an entry in the data set of time series, and r is the search radius. For a range similarity query, $r = \epsilon$. For a k-NN query, on the other hand, the value of r depends both on the value of k and on the data set. In particular, $r = Dist(x_k, q)$, where, from all the entries in the data set, x_k is the kth most similar entry to q.

The cost of answering a similarity search query would be prohibitively high if the query time series had to be compared with a large number of time series in the database. It is therefore extremely important to index the data time series in such a way that relevant time series can be retrieved without looking up the entire database exhaustively. Under traditional approaches, time series data of length l can be mapped into points in an l-dimensional vector space and a spatial access method such as the R-tree [3], [11] can be used to index them. However, it is not uncommon that time series data are sampled during a relatively long period of time. Thus, a direct application of the traditional approaches will require mapping time series data into a very high dimensional vector space and may suffer performance degradation due to reduced pruning power of an index. This phenomenon is known as the *curse of dimensionality*.

To address this problem, many promising techniques have been proposed to reduce the dimensionality of a data

set before using a multidimensional access method. Among those proposed techniques are Discrete Fourier Transformation (DFT) [1], [30], [31], Discrete Wavelet Transformation (DWT) [6], [29], [34], Singular Value Decomposition (SVD) [23], Segmented Means (SM) [36] (also known as Piecewise Aggregate Approximation (PAA) [18]), and Adaptive Piecewise Constant Approximation (APCA) [17]. As pointed out in the APCA work [17], the requirement of the dimensionality reduction is the high fidelity of approximation in an indexable representation.

In this paper, we focus on the problem of whole sequence match and propose a simple and elegant paradigm for indexing time series data. The proposed approach, called *Skyline Index*, adopts new *Skyline Bounding Regions (SBR)* to represent a group of time series data according to their collective shape. The skyline bounding regions allow us to define a distance function that tightly lower bounds the distance between a query object and a group of time series data. We summarize the benefits of building an index based on the skyline bounding regions in the following list:

- *Reducing index access*. Tight lower bounding distances can be defined based on the SBR. This reduces the number of index pages accessed during search. Note that care should be taken during index construction as the insertion of pathological entries (e.g., a time series with a spike) can considerably enlarge an SBR reducing the effectiveness of the lower bounding distances. Of course, this is an extreme case that can arise in any spatial access method.
- *Reducing data access*. A reduction on the number of data objects fetched can result from the combined effect of the lower bounding distance to an SBR and the lower bounding distance to each of the entries in the SBR. We provide theoretical proof and empirical evidence of this effect in Sections 3.5 and 4.4, respectively.
- *Orthogonal to data approximations*. Different approximations can be used to represent data entries and skyline bounding regions. For instance, DFT, DWT, PAA, or APCA approximations can be used to represent data entries in a leaf node. In internal nodes, skyline bounding regions can be approximated either by PAA or APCA.

The rest of this paper is organized as follows: After giving a brief overview of the previous work and its limitations in Section 2, the Skyline Index approach is presented in Section 3. The results of experimental evaluation are given in Section 4. In Section 5, we present some discussions of the Skyline Index with respect to previous work. Finally, Section 6 summarizes the contribution of this paper and gives an outlook to future work.

2 Previous Work

Similarity search on time series data has been extensively studied in the past decade. The focus of research has been on improving performance of similarity matching by reducing the dimensionality of time series data. In this section, we survey the most common techniques to reduce the dimensionality of time series data. To make this paper self-contained and because we make constant use of it, we include a description of the algorithm used to process k-nearest neighbor queries.

2.1 Dimensionality Reduction of Time Series Data

Agrawal et al. [1] proposed the use of Discrete Fourier Transformations (DFT) to represent time series data as vectors in a relatively low-dimensional frequency space. These feature vectors are then indexed using a spatial access method such as the R-tree. They suggested that most of real signals need only a few DFT coefficients for close approximation. This work was later generalized to account for subsequence match queries [10]. Rafiei and Mendelzon [31] proposed an improvement to DFT-based indexing techniques. They observed that the Fourier transform of every real-valued time series is symmetric with respect to its middle. Therefore, they only need to store in the index half of the coefficients used in the distance computation. Chan and Fu [6] suggested the use of the Haar wavelet transformation (a type of DWT) to obtain high-quality representations of time series data. Later, Popivanov and Miller [29] proved that not only orthonormal transformations (such as DFT and Haar), but also biorthonormal wavelets can be used for efficient similarity search on time series data. On the other hand, recent studies by Keogh and Kasetty [19] and Wu et al. [35] have shown that the observed performance gains by using DWT over DFT (or vice versa) seem to depend on the data sets used in the experiments (a phenomenon known as *data bias*). Therefore, it is unclear whether one kind of transformation is better than the other.

Shatkay and Zdonik [33] suggested breaking a time series data into meaningful subsequences and storing approximate and compact representations of the subsequences as mathematical functions. Kahveci and Singh [15] proposed MultiResolution (MR) index for answering variable-length queries for time series data. Under this approach, windows of different sizes (or resolutions) are used to extract subsequences from each data sequence. These subsequences are transformed by either DFT or DWT. The index structure is an $i \times j$ grid, where i is the number of data sequences and j is the number of different window sizes used.

Another approach for reducing dimensionality of time series data is to decrease the resolution of time series. Using piecewise linear segmentations as the underlying representation, Keogh and Smyth [21] suggested a bottom-up approach under which a pair of adjacent segments are merged at each step. The merging process stops when the amount of error introduced by the merging process exceeds a threshold value. Yi and Faloutsos [36] and Keogh et al. [18] independently suggested approximate representation of time series data by dividing it into equal-length segments. Under this approach, called Segmented Means (SM) or Piecewise Aggregate Approximation (PAA), a time series data is approximated by the mean values of the data points within individual segments. More recently, Keogh et al. [17] proposed a new approach, called Adaptive Piecewise Constant Approximation (APCA), which uses variable-length segments in an attempt to better approximate time series data. Since we will be making constant

TABLE 1
Symbolic Notation Used in This Paper

Notation	Description
x	a data time series of length l.
q	a query time series of length l.
x'	the data approximation (feature vector) of the time series x.
$d_{feature}(q, x')$	a generic name for the distance between q and the feature vector x' of x.
$D_{Region}(q, R)$	a generic name for the distance between q and the bounding region R.
$d_{Apca}(q, x')$	the distance from q to the APCA representation x' of x.
$D_{Apca}(q, R)$	the distance from q to the MBR R in the APCA index.
$D_{Sphere}(q, S)$	the distance from q to the sphere S in the M-tree.
$LB_{Keogh}(q, R)$	the distance from q to the SBR R in the Skyline index.

reference to the APCA approximation for the rest of this paper, a more detailed description of this technique follows. For the notation used in this paper, the reader is referred to Table 1.

2.1.1 Adaptive Piecewise Constant Approximation (APCA)

When APCA is used, a time series is approximated by a number of variable-length constant-valued segments, each of which is represented by a pair of its mean value and endpoint in time. The APCA approximation, x', of a time series data, x, is a sequence of $2M$ values in the following format:

$$x' = \{< v_1, r_1 >, \ldots, < v_M, r_M >\}, \quad r_0 = 0,$$

where M is the number of variable-length segments, v_i is the mean value of the data points in the ith segment, and r_i is the right end point of the ith segment. Note that this representation is a generalization of the Segmented Means approach with the restriction of equi-length segments removed. The APCA approximation allows for segments of different size to minimize the approximation error. Figs. 1a and 1c show time series x and y being approximated by APCA using four segments. The min and max values shown here are only used during index construction (i.e., to define MBRs).

In order to use the APCA approximation in a multidimensional index, two new distance functions are used to lower bound the distance between a query and a data time series and the distance between a query and a Minimum Bounding Rectangle (MBR) of data time series. Let us denote the two lower bounding distance functions by $d_{Apca}()$ and $D_{Apca}()$, respectively. The distance, $d_{Apca}(q, x')$,

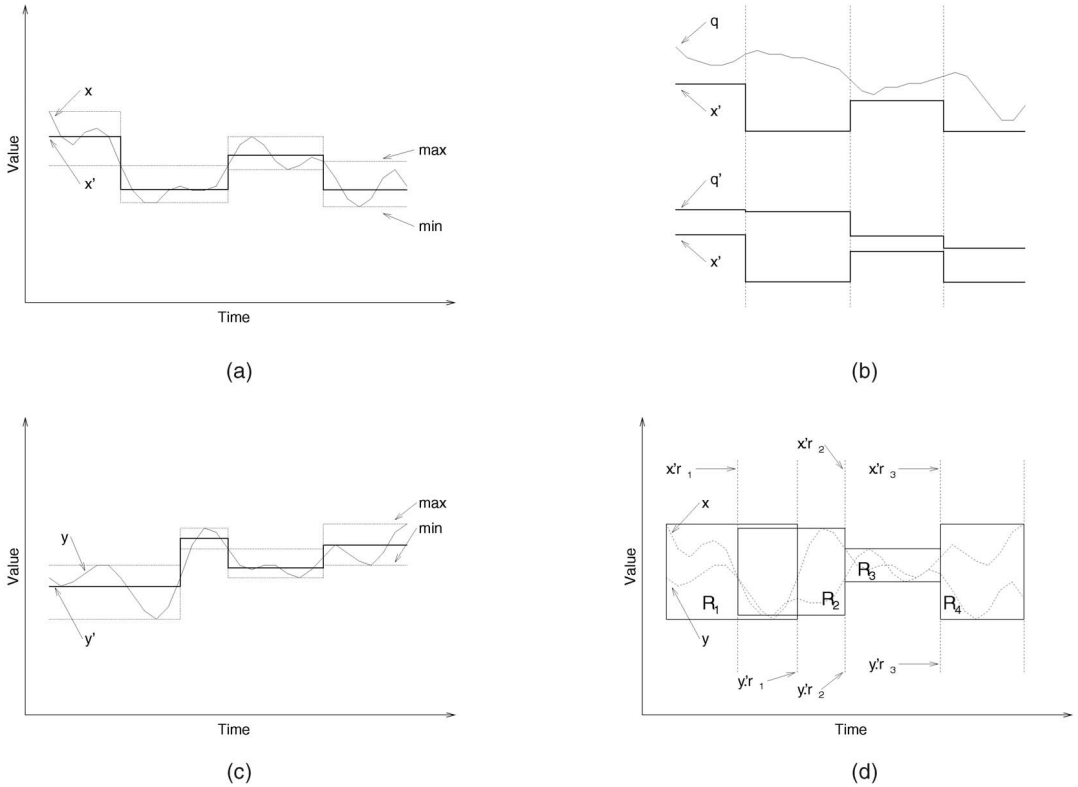


Fig. 1. Adaptive Piecewise Constant Approximation (APCA). (a) The APCA approximation for x. (b) Lower-bounding distance. (c) The APCA approximation for y. (d) APCA regions for x and y.

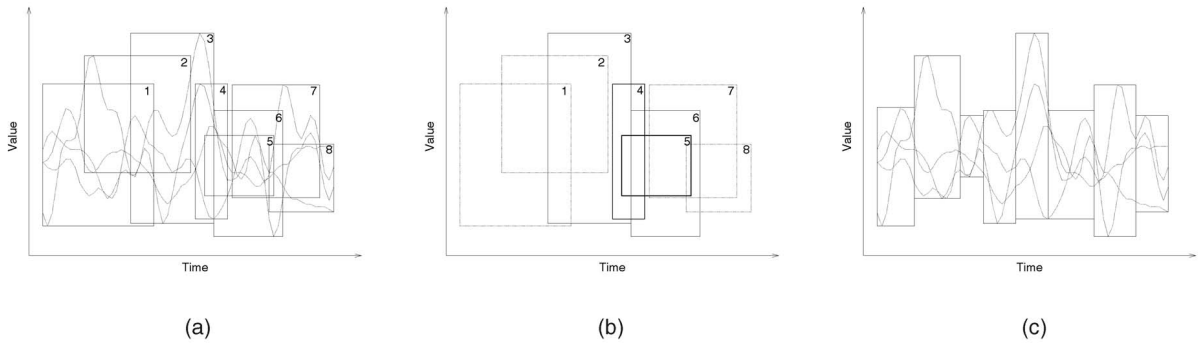


Fig. 2. Limitations of the APCA MBRs. (a) An APCA MBR. (b) Rectangle 5 is redundant. (c) An MBR without overlap.

from a query q to the APCA representation x' of a data time series x is computed by first segmenting q using the same segment boundaries of x'. This process is illustrated in Fig. 1b. Once q has been segmented, a lower bound to the distance between x and q is computed by aggregating the length-weighted differences of the mean values between the segments in q and the segments in x'. The $D_{Apca}()$ function, on the other hand, is computed based on a new definition of MBR. Under the APCA approximation, an MBR R in the $(2M)$-dimensional space logically defines M rectangular regions in a two-dimensional *time-value* space, $R = \{R_1, R_2, , R_M\}$. For an index node U, R_i is the minimum bounding rectangle (in the *time-value space*) containing the ith segments of all the time series data indexed under U. For an index node U, each $R_i \in R$ is defined as

$$
\begin{aligned}
R_i[1] &= min\{x'.min_i | x' \in U\},\\
R_i[2] &= min\{(x'.r_{i-1}+1) | x' \in U\},\\
R_i[3] &= max\{x'.max_i | x' \in U\},\\
R_i[4] &= max\{(x'.r_i) | x' \in U\},
\end{aligned}
$$

where $R_i[1]$ and $R_i[2]$ define the leftmost lower corner of R_i and $R_i[3]$ and $R_i[4]$ define the rightmost upper corner of R_i. In Fig. 1d, we illustrate these regions for the APCA approximations of x and y using four segments. The lower bounding distance $D_{Apca}()$ is computed with respect to the region formed by the union of this set of two-dimensional rectangles.

2.1.2 Limitations of APCA

While the APCA representation is the most promising technique for efficient similarity search of time series data, there is a potential weakness in the method used to define bounding regions during indexing. In particular, the two-dimensional rectangles, $R_1, R_2, \ldots, R_M$, defined by the MBR of an index node can overlap. Overlapping rectangles could have a negative effect on the search performance. This effect can be explained in two different ways. Fig. 2a shows a group of time series data and a set of two-dimensional rectangles defined by an APCA MBR, R. For this example, we have used eight segments in the APCA representation, and the APCA MBR defines eight two-dimensional rectangles. In Fig. 2b, we have eliminated the time series data for better visualization of the two-dimensional rectangles. In this figure, it becomes clear that rectangle 5 is redundant because it is completely covered by rectangles 3 and 6. Therefore, the same shape for R can be defined using less space in the index node (i.e., a smaller key). In this example, we could have reduced the key size by 12.5 percent had we decided to ignore rectangle 5. For this reason, after eliminating redundant rectangles, it would be possible to reduce the index size and, in consequence, improve search performance.

Alternatively, using the same amount of space in the index node, a better MBR could be defined if we use a representation free of overlaps, that is, an MBR that can represent more accurately the collective shape of a group of time series data. This case is illustrated in Fig. 2c, where we have used the same number of rectangles as in Fig. 2a except that care has been taken to avoid overlaps. It is clear that the amount of *dead space* (i.e., the portion of the rectangles not containing data) in Fig. 2c is less than the dead space in Fig. 2a. Therefore, tighter lower bound distances to a group of time series can be defined, resulting in an improved performance for similarity search. Note that we are referring to the overlap among the two-dimensional rectangles, $R_1, R_2, \ldots, R_M$, defined by a $2M$-dimensional APCA MBR R. Let us call this type of overlap *internal overlap*, which is different from the overlap between MBRs (i.e., *external overlap*) commonly observed in spatial access methods.

We conducted a preliminary experiment to verify our intuition on the existence of *internal overlap* in the MBRs defined by the APCA approximation. To quantify the amount of *internal overlap* present in APCA indexes, we define the following indicator:

$$
O(R) = \frac{\sum_{i=1}^{M} Area(R_i)}{Area(\cup_{i=1}^{M}(R_i))},
$$

which is the ratio of the sum of the areas of all the two-dimensional rectangles defined by an APCA MBR R to the area of R. A large $O(R)$ value indicates that the amount of *internal overlap* is considerable and there is room for improvement in the search performance by using an internal overlap-free bounding region.

In this, and all our preliminary experiments, we used a data set composed of 99,000 electroencephalograms (EEG) of length 256. This data set was obtained from the UCI KDD data archive [13]. We built three different indexes using the APCA representation with 8, 16, and 32 segments, respectively (i.e., $\mathrm{M} = 8, 16, \text{ or } 32$). Because each segment requires

TABLE 2
Internal Overlap of the Regions Defined by APCA

Segments (M)	$\Sigma Area(R_i)$	$Area(\cup(R_i))$	$O(R)$
8	1.180	0.515	2.291
16	3.140	0.867	3.622
32	8.679	1.564	5.549

two values, the dimensionality of the approximations was 16, 32, and 64, respectively. We computed $O(R)$ for all leaf nodes in each index and present the average of our measurements in Table 2. This table shows that the sum of the areas of all rectangles was up to 5 times the area of the MBR, indicating a considerable amount of *internal overlap*. We should note that we have purposely ignored the units of the values in columns 2 and 3 of Table 2. The units for both columns are defined in the $time \times value$ domain. However, since we are only interested in the ratio between the two values, their units are irrelevant for our study. These results motivate our approach that groups time series data by new *skyline bounding regions (SBR)* to provide tighter lower bounding distances.

2.2 The k-Nearest Neighbor Search Algorithm

Similarity search and, in particular, k-nearest neighbor (k-NN) search can be implemented using the APCA approximation for time series data. Keogh et al. used a variation of Seidel and Kriegel's optimal multistep k-NN search algorithm [32] and showed that the APCA approximation can be combined with an index for answering similarity search queries [17]. The APCA approximation can be used despite the fact that its lower bound distance function $d_{Apca}()$ does not satisfy the triangular inequality.

We include Keogh's k-NN search algorithm here (Algorithm 1) since we need it to explain some of the concepts presented in this paper. Algorithm 1 works on multidimensional index structures such as the R-tree. The leaf nodes contain approximate representations (feature vectors) of time series data. These feature vectors are used to calculate the lower bounding distance $d_{feature}()$ between a query and a data time series. Each entry in the internal nodes is a minimum bounding region (MBR). An MBR can be a bounding rectangle of the feature vectors of its subtree. It can also be a region defined on the approximations of the actual data contained by the MBR. This internal entry is used to calculate the lower bounding distance $D_{Region}(q, R)$ between a query q and an MBR R. Note that $d_{feature}()$ and $D_{Region}()$ are only generic function names. They are defined depending on the dimensionality reduction techniques used for both data and bounding regions.

The correctness of this k-NN search algorithm (Algorithm 1) with respect to the lower bounding distances provided by the index is summarized by the following lemma:

Lemma 1. *The answer for a k-NN search query using Algorithm 1 is correct if and only if the following two conditions on $d_{feature}()$ and $D_{Region}()$ are satisfied:*

1. *$d_{feature}(q, x') \leq Dist(q, x), \forall\ x$ in the database. This condition is also known as the* **contractive property**,
2. *$D_{Region}(q, R) \leq Dist(q, x), \forall\ x$ in the MBR R. This condition is referred to as the* **group lower bound property**,

where q is a query time series, x' is the data approximation of time series x, and R is an index node's MBR. $Dist(q, x)$ is the distance function used to estimate the similarity between two time series (usually by Euclidean distance). For the rest of this paper, we assume $Dist(q, x) = ||q - x||_2$.

Algorithm 1: K Nearest Neighbor Search
Input: (q, k)
// q is the query object.
// k is the number of nearest neighbors searching for.
1. enqueue(prio_queue, root, 0);
2. **while** *prio_queue is not empty* **do**
3. entry e $\leftarrow$ dequeue(prio_queue);
4. **switch** *the object pointed by e* **do**
5. **case** *a data object*
6. result $\leftarrow$ result $\cup$ {e};
7. **if** $|result| = k$ **then** return result;
8. **case** *an entry in a leaf node*
9. fetch the actual data object x pointed by e;
10. enqueue(prio_queue, x, $||q - x||_2$);
11. **case** *an index leaf node*
12. **for** each data approximation x' in leaf node e do
13. enqueue(prio_queue, x', $d_{feature}(q, x')$);
end
14. **case** *an index internal node*
15. **for** *each child node R in e* **do**
16. enqueue(prio_queue, R, $D_{Region}(q, R)$);
end
end
end

Lemma 1 gives us the guidelines for designing index organizations and dimensionality reduction techniques. The first property is usually supported by feature extraction or dimensionality reduction techniques. The second property should be supported by the indexing technique of choice. Note that, in Lemma 1, there is no mention of the *triangular inequality* about the feature distance $d_{feature}()$. As long as the two conditions are satisfied, Algorithm 1 will yield correct results.

For brevity, we omit the proof of Lemma 1. The interested reader can find a detailed proof of this lemma in the original APCA work [17].

3 Skyline Index Organization

In this section, we describe the *Skyline index*. First, we introduce the *Skyline Bounding Region (SBR)*, which is the core idea of the proposed approach. Then, we describe the fundamental details regarding a lower bounding distance function for SBRs and the adoption of SBRs for indexing and searching time series data.

3.1 Skyline Bounding Regions

Under traditional methods for dimensionality reduction, time series data are first mapped into points in a low-dimensional feature space and the points (i.e., feature

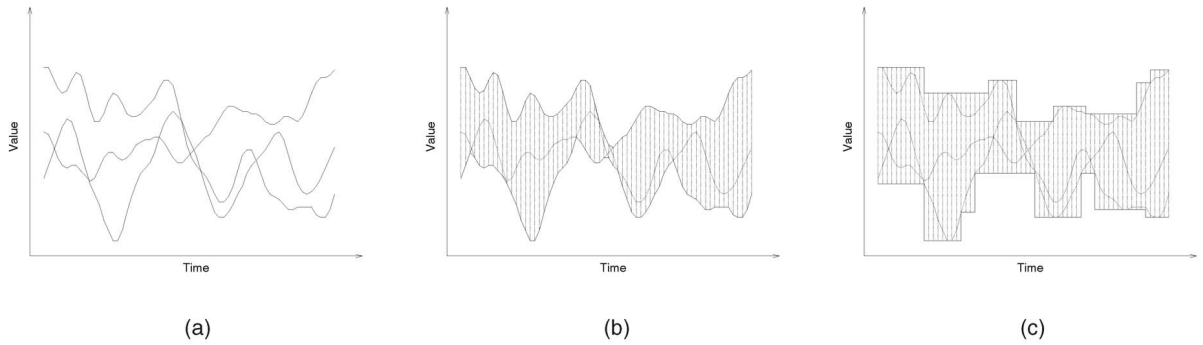


Fig. 3. Skyline bounding region of time series data. (a) Three time series. (b) Skyline bounding region. (c) Approximate SBR.

vectors) are then grouped into MBRs to be stored in a hierarchical index structure. In this section, we present an alternative way to organize time series data, which enables the dimensionality reduction techniques to maintain high fidelity of approximation, and thereby improving the performance of similarity query processing.

A time series data exists in a two-dimensional *time-value* space. If a minimum bounding rectangle (MBR) defined in this space is used to bound a group of time series data, the external overlap between MBRs will be large, and the search performance will suffer from this. In order to minimize the overlap between MBRs, it is desirable to approximate a group of time series data with tighter bounding regions. To achieve this goal, we propose using *skylines* to bound a group of time series data. This bounding region, called *Skyline Bounding Region (SBR)*, is defined as follows:

Definition 1 (Skyline Bounding Region (SBR)). *For a given group S of n time series data objects of length l, $S = \{s_1, s_2, ..., s_n\}$, the Skyline Bounding Region of S specifies a two-dimensional region surrounded by top and bottom skylines and two vertical lines connecting the two skylines at the start and end times. The top (TSky) and bottom (BSky) skylines of S are defined as follows:*

$$TSky = \{ts_1, ts_2, \ldots, ts_l\}, \quad BSky = \{bs_1, bs_2, \ldots, bs_l\},$$

where, for $1 \le i \le l$,

$$ts_i = max\{s_1[i], \ldots, s_n[i]\} \text{ and } bs_i = min\{s_1[i], \ldots, s_n[i]\},$$

and $s_j[i]$ is the ith value of the jth time series data in S.

Note that, unlike MBRs generated from the APCA representation, Skyline Bounding Regions are composed of only one region. Therefore SBRs are free of internal overlap. As an example, Fig. 3b shows the skyline bounding region for the three time series data in Fig. 3a. Evidently, the straightforward adoption of the SBR definition is impractical because its representation can be too costly for long time series data. Therefore, only approximate representations of $TSky$ and $BSky$ should be stored for each SBR in an index structure. Care should be taken to ensure that the region defined by the approximate SBR encloses that of the original SBR. This will ensure the *group lower bound property* (second condition of Lemma 1) is met by the skyline distance function, which will be introduced next. Two good candidates for approximating the skylines are to use either equal-length or variable-length constant-valued segments, in a similar way to Segmented Means [36] and Adaptive Piecewise Constant Approximation [17]. Among these candidates, it has been shown that the APCA representation has the smallest approximation error [17]. Thus, we have chosen to use APCA to approximate SBRs in the Skyline index. An example of variable-length constant-valued segments approximating a skyline bounding region is shown in Fig. 3c. For the rest of this paper, we assume the APCA representation is used to approximate Skyline Bounding Regions.

3.1.1 Quality of the Bounding Regions

In this section, we compare the Skyline index with the APCA index for the quality of indexing. We base our comparisons on the differences between the bounding regions defined by the indexes (*index bounding regions*) and the bounding regions defined by raw data (*data bounding regions*). If we think of a time series data as an l-dimensional vector, the *data bounding region* for a group, G, of l-dimensional vectors is the minimal l-dimensional bounding hyperrectangle for G. We estimate the quality of the index by the ratio of the area covered by the *index bounding region* (in the *time-value* space) to the area covered by the *data bounding region* as follows:

$$Q(I_R) = \frac{Area(IndexBoundingRegion)}{Area(DataBoundingRegion)}, \tag{1}$$

where I_R is an index node for G, $IndexBoundingRegion$ is the bounding region, R, defined by the index for node I_R, and $DataBoundingRegion$ is the bounding region defined by G. For the Skyline index, this ratio is defined as

$$Q_{skyline}(I_R) = \frac{Area(ApproximateSBR)}{Area(DataBoundingRegion)}.$$

For the APCA index, we use

$$Q_{Apca}(I_R) = \frac{Area(\cup_{i=1}^{M}(R_i))}{Area(DataBoundingRegion)}, \quad \{R_1, R_2, \ldots, R_M\} = R,$$

where R is the set of M two-dimensional rectangles defined by the MBR for node I_R in the APCA index.

TABLE 3
Quality of Bounding Regions

Segments	$Q_{Apca}(I_R)$	$Q_{skyline}(I_R)$
8	1.673	1.326
16	1.70	1.345
32	1.70	1.333

In our preliminary experiments with the EEG data set, we computed these ratios for both skyline and APCA bounding regions. For each method, we built three indexes using 8, 16, and 32 segments (i.e., 16, 32, and 64 dimensions), respectively. In this experiment, both the APCA and the Skyline index are based on the R-tree. We traversed each index and computed the quality of the bounding region for each index node. Table 3 shows the average results from our comparisons. We observed that the area of a bounding region as defined by the APCA index is 70 percent larger than the bounding region defined by the raw data. The Skyline index, on the other hand, defined smaller bounding regions, only 33 percent larger than the regions defined by the raw data. Remember that the bounding region defined by raw data is minimum. Therefore, we used the area of the data bounding region as a yardstick in this experiment. These results suggest that the use of skyline bounding regions will result in a more efficient index. In Fig. 4, we show an example of data bounding region (Fig. 4a), APCA index bounding region (Fig. 4b), and Skyline index bounding region (Fig. 4c). From this figure, it becomes clear that the skyline technique provides a more accurate representation of a data bounding region than the APCA.

3.2 Skyline Distance Function

In order to use the SBR representation in a multidimensional index, we must have a distance function $D_{Region}()$ that lower bounds the distance between a query object and a group of time series data. Keogh [16] defined a distance function $LB_{Keogh}()$ to lower bound the Dynamic Time Warping distance between a data time series and an envelope containing all possible temporal shiftings of the query time series given a constraint on the warping path. In the Skyline index, we have a set of data time series contained within an SBR (a kind of envelope). Therefore, we can use $LB_{Keogh}()$ to lower bound the distance from a query object to an SBR. Formally, given a query q and an SBR R (e.g., an index node) that contains a group of time series data, the lower bounding distance function $LB_{Keogh}()$ is defined as follows:

$$LB_{Keogh}(q,R) = \sqrt{\sum_{i=1}^{l} (dist(q[i], TSky[i], BSky[i]))^2}, \quad (2)$$

where $q[i]$ is the ith value of the query q, l is the length of each time series data, and

$$dist(q_i, t_i, b_i) = \begin{cases} q_i - t_i & \text{if } q_i > t_i \\ b_i - q_i & \text{if } q_i < b_i \\ 0 & \text{otherwise.} \end{cases}$$

The following lemma shows that $LB_{Keogh}()$ satisfies the *group lower bound property*.

Lemma 2. *Given a query q and any time series data x contained in an SBR R,*

$$LB_{Keogh}(q,R) \leq ||q - x||_2, \forall x \in R, \quad (3)$$

where $||q - x||_2$ represents the Euclidean distance between q and x.

Proof. For any $x \in R$, a value $x[i]$ at time $i\,(1 \leq i \leq l)$ is bounded by $TSky[i]$ and $BSky[i]$. Thus,

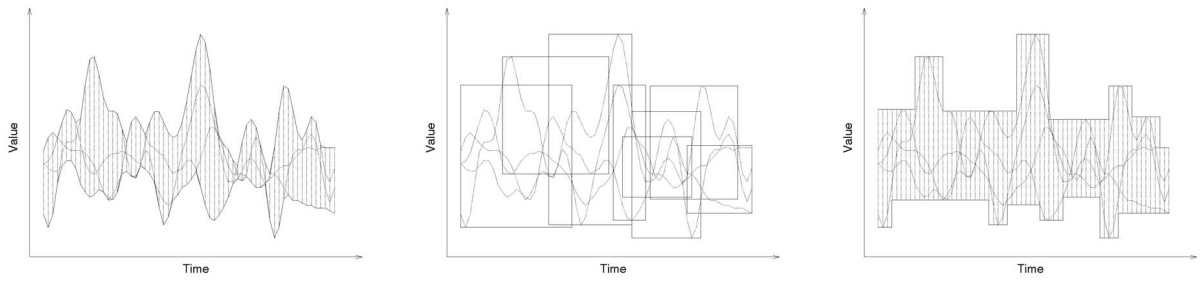
$$(dist(q[i], TSky[i], BSky[i]))^2 \leq (q[i] - x[i])^2.$$

From this, it is obvious that $LB_{Keogh}(q,R)$ satisfies the group lower bound property. □

Note that (2) and Lemma 2 have been described based on full SBR representations (e.g., Fig. 3b) for simpler presentation. The $LB_{Keogh}()$ distance function can also be defined for approximate SBR representations (e.g., Fig. 3c) and Lemma 2 will still be satisfied.

3.3 Indexing Time Series

Following the GEMINI [8] paradigm, the approximate representations of time series data can be indexed by a spatial access method. We implemented the Skyline index based on the R-tree structure [11]. In an R-tree based *Skyline index*, each entry in an internal node consists of the approximate representation of an SBR and a pointer to a child node. An entry in a leaf node, on the other hand, consists of the approximate representation of and a



(a) (b) (c)

Fig. 4. Data and Index Bounding Regions (IBR). (a) Data Bounding Region. (b) APCA IBR. (c) Skyline IBR.

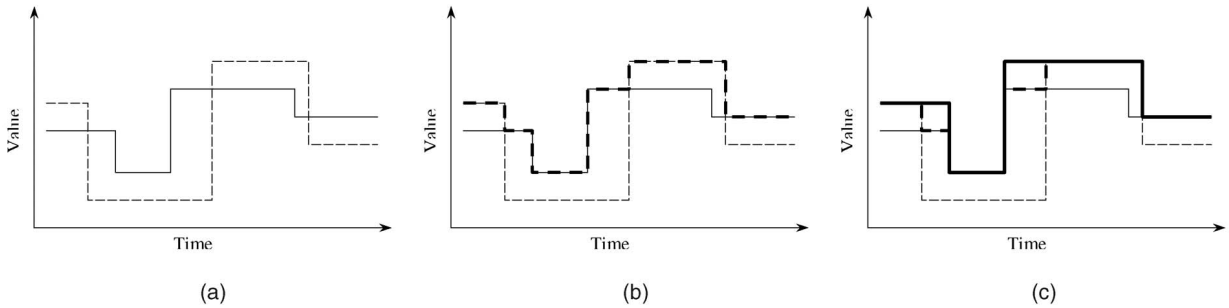


Fig. 5. Merging two skylines. (a) Two top skylines. (b) Intermediate top skyline. (c) Resulting top skyline.

pointer to a time series data object. Note that the *Skyline index* is not restricted to a particular approximation to represent time series data objects. Since it is possible to calculate the SBR of a leaf node based on raw time series data, we can use any approximation method as long as the corresponding $d_{feature}()$ function satisfies the *contractive property* (condition 1 in Lemma 1). Alternatives for approximating time series data are to use variable-length constant-valued segments, equal-length constant-valued segments, or even DFT or DWT transformations. We have chosen to use variable-length constant-valued segments (i.e., APCA) to represent time series data because of the high fidelity of its approximation [17].

The insertion algorithm used by the *Skyline index*, which is described below, is similar to the R-tree insertion algorithm proposed by Guttman [11].

Step 1. *Find a position for a new record*. Starting from the root node, traverse the tree to find the best leaf node for inserting the new entry.

Step 2. *Add the record to a leaf node*. Split the node if it is full.

Step 3. *Propagate changes upward*. Ascend the tree from the leaf node. Adjust SBRs and propagate node splits as necessary.

Step 4. *Grow the tree taller*. If propagation caused the root to split, create a new root whose children are the two resulting nodes.

During **Step 1**, at each level of the tree, *Skyline index* selects the node whose approximate SBR requires the least area enlargement to accommodate the new entry. An approximate SBR is enlarged if, for a given segment, a value in the new entry is outside the limits defined by the segment. In such a case, the value of the segment is updated to completely contain the new entry. At **Step 2**, if a split is needed, entries are redistributed by Guttman's quadratic algorithm [11] to minimize the area of the resulting SBRs.

In a hierarchical index structure, the SBR of a parent node is determined from the SBRs of its child nodes such that the parent SBR minimally encloses all the child SBRs. In **Step 3**, some SBRs may need modifications to accommodate a new entry. Here, we describe how to expand the SBR of an internal node. Later, we describe how to obtain the SBR of a leaf node during insertion. An SBR consists of a top and bottom skyline. Since expanding the bottom skyline is symmetrical, we only focus on describing how the top skyline is expanded. We start by merging the top skylines of the approximate SBR of an internal node and a new entry. Fig. 5a shows an example of two skylines to be merged. Each skyline has four segments. We scan the two skylines together from left to right along the time dimension, generating a new intermediate top skyline (Fig. 5b). If the resulting number of segments is larger than required, we iteratively coalesce pairs of consecutive segments until we get the required number of segments. Two consecutive segments, $< v_i, r_i >$ and $< v_{i+1}, r_{i+1} >$, are coalesced into $< v'_i, r_{i+1} >$, where $v'_i = max\{v_i, v_{i+1}\}$. Note that, in a top skyline segment $< v, r >$, v is the maximum value and r is the right end of the segment. The pair of consecutive segments that results in the minimum increase of the approximation error is chosen in each coalescing operation. Fig. 5c, shows the final result from merging the skylines in Fig. 5a. After coalescing, only four segments are left in the resulting skyline.

One concern about iteratively coalescing pairs is that the quality of the approximate SBR for an index node (U) may degrade after consecutive insertions. Consequently, the resulting approximate SBR, which we refer to as the *merge-approximate SBR*, may not tightly approximate the SBR defined by the time series data indexed by U. We conducted one more preliminary experiment using the EEG data set to investigate this issue. We built three Skyline indexes using 8, 16, and 32 segments, respectively. Each index was incrementally built by inserting 99,000 entries. For each index node U, we fetched all time series data indexed by the subtree rooted at U. The collective shape of this set of time series data defines the SBR for U. We approximated this SBR (e.g., using APCA), resulting in the *direct-approximate SBR* for U. Note that this direct-approximate SBR was generated from the SBR defined by raw time series data. Therefore, it is free of any quality degradations due to the merging mechanism used to process insertions. We compared the direct-approximate SBR of U to its *merge-approximate* SBR (i.e., the approximate SBR for U stored in the index). To measure the degradation of the index SBR due to the merging process, for each index node, we computed the quality of both its *direct-approximate* and *merge-approximate* SBR using (1) given in Section 3.1.1. We present the average results of our measurements in Table 4, labeled as Q_{direct} and Q_{merge}, respectively. These results show that the quality degradation of the approximate SBRs after consecutive insertions is insignificant (i.e., less than 5 percent) with respect to the direct-approximate SBR. This gave us an indication that the proposed merging mechanism is a viable technique to incrementally maintain approximate SBRs in the Skyline index. We should note that we used the same

TABLE 4
Quality of Merge-Approximate SBR versus Direct-Approximate SBR

Segments	Q_{merge}	Q_{direct}	Q_{merge}/Q_{direct}
8	1.326	1.329	0.998
16	1.345	1.326	1.014
32	1.333	1.280	1.041

suboptimal implementation of the APCA as in [17] to generate the direct-approximate SBRs. This had an adverse effect on the quality of the approximation, which was more evident for approximations using a small number of segments. As is shown in the first row of Table 4, when only eight segments are used, the merging error introduced by the index was smaller than the error introduced by the suboptimal approximation.

3.3.1 Approximate SBRs in Leaf Nodes

We have mentioned before that the Skyline index is not restricted to a specific data approximation to represent time series. For example, we could approximate time series data using variable-length constant-valued segments, equal-length constant-valued segments, or even DFT or DWT transformations. Regardless of the data approximation of choice, we can trivially generate the SBR for a leaf node by retrieving all the time series data indexed by the node. Once we have obtained the SBR, we can approximate it and then merge it into the index as described before (i.e., Step 3 of the insertion algorithm).

An evident problem with this approach is a poor performance due to the excessive IO required by each insertion (inserting a new entry in leaf node U requires fetching all time series data contained in U). Instead, we adopt an alternative solution in which a new time series data entry is considered to be an SBR. A data time series of length l is an SBR with l segments in which its top and bottom skylines are equal. Because of the high cost of merging l segments, we opt for merging an approximation of the time series using a smaller number of segments. The approximation of this SBR can be easily obtained. For example, we could use the APCA approximation in which case the min and max values of each segment define the top and bottom skyline of the new entry. After this, expanding a leaf node SBR is done in the same way the SBR for an internal node is expanded (described before).

3.4 Skyline k-Nearest Neighbor Search

The Skyline index can be used with Algorithm 1 to provide k-Nearest Neighbor (k-NN) Search. We need to substitute the generic functions $d_{feature}()$ and $D_{Region}()$. If we choose APCA as the approximation for time series data, the function $d_{Apca}()$ replaces $d_{feature}()$. The $d_{Apca}()$ distance function lower bounds the Euclidean distance between a query and a data time series. Therefore, Condition 1 of Lemma 1 is satisfied. Also, $D_{Region}()$ will be replaced by $LB_{Keogh}()$, which guarantees the *group lower bound property* of Lemma 1. Thus, it is easy to see that Algorithm 1 using Skyline index will produce correct answers.

3.5 Lower-Bound Distances and Search Performance

In this section, we state the theoretical basis for the argument that the use of skyline bounding regions improves the k-NN search performance. In particular, we investigate the combined effect of the lower bound distance function and the index bounding regions on the number of candidates retrieved during k-NN search. We also study how these factors are related to the concept of r-optimality defined by Seidl and Kriegel [32].

An r-optimal multistep k-NN algorithm must retrieve every candidate object c whose feature distance from a given query q is less than or equal to the Euclidean distance to the kth nearest neighbor x_k from q. That is, $d_{feature}(q,c) \leq ||q - x_k||_2$, where $d_{feature}()$ is a lower bounding distance function adopted by the multistep k-NN algorithm. Let us call the set of all these objects retrieved by the r-optimal algorithm the *minimal candidate set*. This set is *minimal* because it *only* includes objects satisfying $d_{feature}(q,c) \leq ||q - x_k||_2$. It is generally assumed that the distance to a region R lower bounds $d_{feature}(q,x)$, $\forall x \in R$. We formally define this property as the *containment property*.

Definition 2 (Containment Property). *For a given query q and a bounding region R (e.g., an index node) containing a group of time series data, $D_{Region}()$ and $d_{feature}()$ satisfy the* containment property *if the following condition holds.*

$$D_{Region}(q,R) \leq d_{feature}(q,x), \forall x \in R, \tag{4}$$

where $D_{Region}(q,R)$ represents a lower bounding distance from q to the bounding region R and $d_{feature}(q,x)$ is a lower bound of the distance from q to x.

While the containment and the contractive (Lemma 1) properties are necessary conditions for a k-NN search algorithm to be r-optimal, the containment property is not a necessary condition for the *correctness* of the k-NN search algorithm. For instance, it has been shown that the APCA representation produces correct k-NN search results despite the fact that it does not satisfy the containment property [17]. The APCA representation does not satisfy the containment property because the $D_{Apca}()$ distance function is computed based on a definition of MBR, which is independent of the lower bounding distance function $d_{Apca}()$. Specifically, for a bounding region R, while $D_{Apca}(q,R) \leq ||q - x||_2$ and $d_{Apca}(q,x') \leq ||q - x||_2$ $\forall x \in R$, there may exist an $x \in R$ such that $d_{Apca}(q,x') < D_{Apca}(q,R)$. Here, x' is the APCA data representation of x. This is illustrated in Fig. 6, which shows an example of an APCA MBR with one segment. In the figure, the mean values of time series q and x are the same. That is, $d_{Apca}(q,x') = 0$. Since q is not enclosed by the MBR R, $D_{Apca}(q,R) > 0$. Therefore, it is the case that $D_{Apca}(q,R) > d_{Apca}(q,x')$. Clearly, this example illustrates that the *containment property* is not satisfied for the APCA representation.

For a similar reason, the containment property does not hold for $d_{Apca}()$ and $LB_{Keogh}()$ under the proposed Skyline index approach and, hence, Algorithm 1 is not r-optimal either. However, it is still true that the search result will be correct because Lemma 1 holds for $d_{Apca}()$ and $LB_{Keogh}()$. We now show that the number of objects retrieved by this

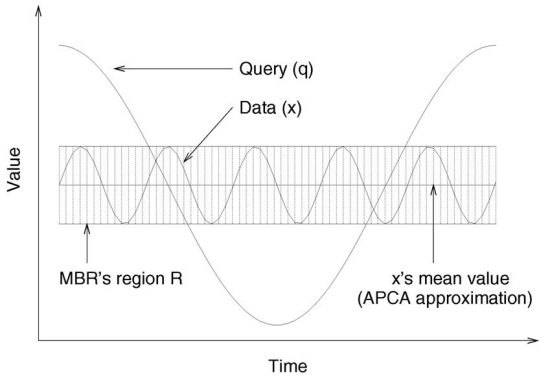


Fig. 6. APCA does not satisfy the containment property.

algorithm could actually be less than the number of elements in the *minimal candidate set* defined by $d_{Apca}()$ (the candidate set is minimal with respect to a lower bounding distance function, changes to this function affect the size of the candidate set). Let us start by defining a *virtual* lower bounding distance to be used by the Skyline index. Given a query q, a data time series x, and its APCA approximation x', we define a new lower bounding distance function $d^v_{Apca}(q, x')$ as follows:

$$d^v_{Apca}(q, x') = \max\{d_{Apca}(q, x'), LB_{Keogh}(q, R)\}, \quad (5)$$

where R is the immediate bounding region (i.e., direct parent) of x. Because $d^v_{Apca}(q, x')$ and $LB_{Keogh}(q, R)$ now satisfy the containment property, Algorithm 1 becomes r-optimal with $d_{Apca}(q, x')$ replaced by $d^v_{Apca}(q, x')$ in Line 13. In general, (5) can be defined as

$$d^v_{feature}(q, x') = \max\{d_{feature}(q, x'), D_{Region}(q, R)\}. \quad (6)$$

Lemma 3 (Interchangeability). *In a multistep k-NN search algorithm (e.g., Algorithm 1), two lower bounding distance functions $d_{feature}(q, x')$ and $d^v_{feature}(q, x')$ as defined in (6) can be used interchangeably without affecting the correctness of results and the number of candidates to be retrieved.*

Sketch of Proof. First, with either distance function, the k-NN algorithm described in Algorithm 1 will terminate when $||q - x_k||_2$ is smaller than the lower bound distance at the front of the priority queue, where x_k is the kth nearest neighbor. We can then prove the lemma by showing that the following assertions are satisfied at the termination of the algorithm: 1) Any candidate retrieved by the algorithm with $d_{feature}()$ is also retrieved by the algorithm with $d^v_{feature}()$ and 2) any candidate retrieved by the algorithm with $d^v_{feature}()$ is also retrieved by the algorithm with $d_{feature}()$. □

Since the virtual function $d^v_{feature}(q, x')$ is always larger than or equal to $d_{feature}(q, x')$, the minimal candidate set defined by the virtual function can be smaller than that defined by $d_{feature}(q, x')$. The implication of Lemma 3 is exceedingly important. It provides the theoretical basis of the argument that the use of *skyline bounding regions (SBR)* can reduce the number of candidates to be retrieved for k-NN search. By Lemma 3, as long as the group lower bound property is satisfied, different techniques can be used to group data into hierarchical index structures. Instead of using the default grouping method provided by the R-tree, one can choose the best grouping technique with tighter bounds, while keeping the correctness of the k-NN search algorithm. Thus, if one can represent time series data with tighter bounding regions than those being used, either by adopting a different approximation or an entirely different index organization, the overall performance of k-NN search can be improved by reducing the number of index accesses and the number of candidates to be retrieved. This is precisely the way we can improve k-NN query processing by adopting the skyline bounding regions in this paper.

4 Experiments

In this section, we empirically demonstrate the improved performance of the proposed *Skyline Index* approach over state of the art techniques such as APCA. In addition, we observe the impact of different distance functions by different dimensionality reduction and indexing techniques on the overall similarity search performance. We performed our comparisons in the context of k-NN similarity search. We studied combinations of the Haar wavelet transformation (DWT) with the M-tree [7] and R-tree. We performed the same study for the APCA data approximation and included the Hybrid-tree [5], in addition to the M-tree and R-tree. During our empirical evaluation, we used three different metrics to assess the performance of each of the techniques included in this study. In particular we used *index search overhead*, *number of data objects fetched*, and *total elapsed time*.

4.1 Data Sets

We collected time series data from various sources of real-world applications and synthetic data generator and placed them into four separate data sets. Then, for each data set, we created three different cases by chopping each time series data into segments of length 1,024, 512, or 256. Each resulting data set contains 100,000 time series objects of the same length. We randomly extracted 1 percent of the entries from each data set. This subset of 1,000 time series objects became our query set. We did this to avoid exact matches with queries in our experiments. In addition, this practice allowed us to use a query object that is in the same domain as the data set. Details of our testing data are given next.

Mixed S1. This data is generated from several data sources such as Space Shuttle data, Arrhythmia, Random Walk, and Exchange rate. They have different characteristics in shape and structure noise [17]. A sliding window of Step 1 was used to chop each long time series data into segments. The segmented time series data were normalized to have a mean of zero and standard deviation of one. This data set contains 100,000 segmented time series.

Mixed S10. This data set is identical to **Mixed S1** except that a sliding window of Step 10 was used to generate segments.

ECG. This data set is the electrocardiogram data from the MIT-BIH database distribution [24]. This data set contains 100,000 segmented time series data generated by a sliding window of step one.

Synthetic. This data set contains a set of stock data synthetically generated by a financial time series benchmark [14]. The synthetic data represent stock values at opening and closing times as well as the highest and lowest value of each day. We generated stock values for 100,000 companies for a period of 256, 512, and 1,024 days.

4.2 Performance Metrics

In our experiments, we evaluated the efficiency of different techniques using three metrics. We measured *index search overhead* and the number of *data objects fetched* as the two main factors affecting overall performance of similarity search. In addition, we measured *elapsed time* as the performance metric directly perceived by the user.

Index search overhead. This is the number of index pages accessed during k-NN search. Ideally, during search, only a small fraction of the index pages are retrieved. Unfortunately, due to the curse of dimensionality, the effectiveness of an index degrades as the dimensionality increases.

Data objects fetched. It has been suggested that we should prevent performance bias due to disparities in the quality of implementation of the methods being compared [19]. Following this recommendation, we evaluated the effectiveness of different k-NN search methods by the number of *data objects fetched* from the database. The number of data objects fetched during a k-NN query represents a performance metric that is independent of the quality of implementation.

Elapsed time. We used wall-clock time to measure the elapsed time during the evaluation of k-NN queries. This time includes both CPU and IO time. In addition, for each query, we recorded the time spent on CPU operations only. This allows us to estimate the time spent in IO operations. Finally, to avoid any caching effects on the index and data files, we flushed the main memory between consecutive queries by loading irrelevant data into the system.

For each presented measurement, we ran 100 queries and averaged the results from those 100 runs. Throughout the experiments, we measured the performance of k-nearest neighbor queries processed by different index techniques.

4.3 Evaluated Techniques

To show the performance advantages of our proposed Skyline index, we compare it to some of the widely accepted techniques for k-NN search for time series data. Recent research work indicates that the choice of Discrete Fourier Transform (DFT) over Discrete Wavelet Transform (DWT) for similarity search seems to be data-dependent [19]. We have considered this observation and conducted empirical comparisons between these two techniques. We observed that, in three of our data sets, DWT was between 8 and 70 percent faster than DFT for answering k-NN queries. For the other data set, DFT was about 25 percent faster than DWT. For the rest of our experiments, we used DWT as the representative of these two dimensionality reduction techniques. We also used APCA for reducing dimensionality of time series data. To organize these representations, we built indexes based on the H-tree, R-tree, and M-tree.

Hybrid-tree. Since the Hybrid-tree [5] was used to empirically evaluate the APCA in its original work [17], we used the same Hybrid-tree implementation to accurately replicate the work. However, this implementation builds indexes in memory and can only be used to simulate disk IO by counting node accesses. Given our interest in measuring the actual performance of similarity queries including the time spent on disk IO, we used the hybrid tree only for APCA and just counted IO operations. The source code of the hybrid tree was obtained from http://www-db.isc.uci.edu.

R-tree. The Skyline index was implemented using the R-tree interface provided by the GiST [12] library. We extended the R-tree GiST interface and provided two new objects; *Skyline Bounding Region* and *APCA point*. Skyline bounding regions are used for handling approximated SBRs in internal nodes, whereas APCA points are used for handling data approximation in leaf nodes. The actual implementation of the Skyline index using a popular index library such as GiST demonstrates the practical feasibility of the idea of using skyline bounding regions for indexing time series data. We also used GiST for implementing the R-tree index in combination with the APCA and DWT approximations. This gave us a common implementation framework for fair performance comparisons.

M-tree. We include the M-tree in this study to provide a framework for comparing different index organizations and to illustrate the effect that different bounding regions have in the k-NN search performance. In the M-tree, a bounding region S is defined by a hyper-sphere centered at a routing object o_c with a radius γ. The radius γ is the maximum Euclidean distance of any object in S from the routing object o_c (i.e., $||s - o_c||_2 \leq \gamma,\ \forall s \in S$). Thus, for any time series data $s \in S$ and a query q,

$$\begin{aligned} ||q - s||_2 &\geq ||q - o_c||_2 - ||s - o_c||_2 \\ &\geq ||q - o_c||_2 - \gamma \\ &\geq d_{Apca}(q, o_c) - \gamma \end{aligned}$$

because $d_{Apca}(q, o_c)$ lower bounds $||q - o_c||_2$. Therefore, we can define a lower bounding distance for S as follows:

$$D_{Sphere}(q, S) = max\{d_{Apca}(q, o_c) - \gamma, 0\}. \tag{7}$$

Now, let us describe how different indexing mechanisms were combined with the dimensionality reduction techniques in our experiments. In the following, we summarize all the techniques we evaluated. In the description, we use q and x to represent query and data time series, respectively, and R to represent a bounding region. We use x' to denote the data approximation of x.

Skyline. We implemented the Skyline index based on the R-tree (implemented by the GiST C++ library [12]). The $LB_{Keogh}(q, R)$ was used to lower bound

$$\{||q - x||_2 \mid \forall x \in R\}.$$

The APCA representation was used to approximate individual time series data and, accordingly, $d_{Apca}(q, x')$ was used to lower bound $||q - x||_2$. We approximated time series data in APCA representations following the steps suggested in [17].

TABLE 5
IO Performance for 1-NN Queries

Dim	Number of Data Objects Fetched						Number of Index Pages Accessed					
	H-tree APCA	R-tree APCA	R-tree DWT	M-tree APCA	M-tree DWT	Sky-line	H-tree APCA	R-tree APCA	R-tree DWT	M-tree APCA	M-tree DWT	Sky-line
ECG Data (1024)												
16	158	156	3,678	157	3,678	155	2,489	3,759	1,137	4,239	4,239	3,086
32	10	8	565	9	565	8	4,938	6,379	1,350	7,277	7,278	3,150
64	3	2	48	2	48	2	10,457	10,842	1,210	14,593	14,621	2,033
Mixed S10 Data (1024)												
16	3,270	2,480	1,103	3,133	1,103	1,708	2,444	2,415	1,223	1,527	1,396	804
32	1,819	1,462	1,010	1,679	1,010	863	4,560	4,138	2,359	2,277	2,141	1,156
64	1,299	998	986	1,218	986	439	9,887	7,983	4,669	3,957	3,789	1,665
Synthetic Data (1024)												
16	1,915	1,914	94	1,914	94	1,913	2,500	1,670	1,875	2,292	2,200	1,345
32	78	77	17	77	17	77	4,947	3,409	3,874	3,845	3,771	2,259
64	14	14	5	14	5	14	10,500	6,806	7,496	6,900	6,833	4,142

TABLE 6
IO Performance for 10-NN Queries

Dim	Number of Data Objects Fetched						Number of Index Pages Accessed					
	H-tree APCA	R-tree APCA	R-tree DWT	M-tree APCA	M-tree DWT	Sky-line	H-tree APCA	R-tree APCA	R-tree DWT	M-tree APCA	M-tree DWT	Sky-line
Mixed S1 Data (256)												
16	26,292	17,769	15,890	25,907	15,890	12,072	2,421	2,636	1,798	3,162	2,836	1,524
32	20,582	11,489	8,232	20,095	8,232	6,332	4,918	4,617	3,478	4,886	4,145	2,292
64	15,859	8,487	5,090	14,980	5,090	2,970	10,553	9,382	6,407	8,653	7,172	3,333
Mixed S1 Data (512)												
16	24,866	15,185	16,928	24,528	16,928	11,077	2,536	2,538	1,685	2,964	2,670	1,531
32	19,793	10,764	9,944	19,419	9,944	7,089	5,010	4,535	2,866	4,637	4,001	2,254
64	14,503	7,925	5,772	14,054	5,772	3,460	10,578	9,484	5,310	8,089	6,750	3,374
Mixed S1 Data (1024)												
16	24,900	15,079	22,714	24,659	22,714	10,423	2,521	2,460	1,175	2,768	2,676	1,505
32	22,954	11,686	20,022	22,899	20,022	7,805	4,990	4,398	2,384	4,648	4,382	2,157
64	20,881	10,532	13,785	20,797	13,785	4,940	10,708	9,479	4,819	8,353	7,393	3,479

Hybrid-tree APCA. This is the same implementation as described in [17]. The Hybrid-tree [5] was used to index time series data in the APCA representation. The $d_{Apca}(q, x')$ and $D_{Apca}(q, R)$ were used to lower bound $||q - x||_2$ and $\{||q - x||_2 \mid \forall x \in R\}$, respectively.

R-tree APCA. Same as **Hybrid-tree APCA** except that the GiST R-tree was used instead of the Hybrid-tree.

M-tree APCA. The distance function $D_{Sphere}()$, defined by (7), was used in M-tree. Note that $d_{Apca}()$ does not satisfy the triangular inequality [17]. Therefore, we cannot build an M-tree based on such distance. To overcome this problem, we measured the radius of a hyper-sphere using the Euclidean distance defined by raw, instead of approximated, time series data (but only data approximations were inserted into the index). The $d_{Apca}(q, x')$ distance function was used to lower bound $||q - x||_2$.

R-tree DWT. We implemented the Haar wavelet transformation [6] and indexed it using the GiST R-tree index.

M-tree DWT. The Haar wavelet transformation and its feature distance function were used with the M-tree index. To make it comparable to **M-tree APCA**, we measured the radius of a hyper-sphere by Euclidean distance, and used a distance function similar to $D_{Sphere}(q, R)$ to lower bound $\{||q - x||_2 \mid \forall x \in R\}$.

All indexes were built using pages of 8 KBytes. All values in time series data were stored in 8-byte long `double` format. Testing and benchmarking were performed on Intel Pentium workstations running Linux kernel version 2.4.7. Each workstation has 600 MHz clock rate, 128 MBytes of main memory, and 9 GBytes of disk storage with SCSI interface.

4.4 Experimental Results

In our experiments, we extensively evaluated the performance of the different k-NN search techniques described in Section 4.3. We measured the performance of these techniques using different data sets, different values of k, and different lengths of the time series.

4.4.1 Index Search Overhead and Data Objects Fetched

Tables 5 and 6 show the IO performance results obtained from our experiments. Each table shows the number of *data*

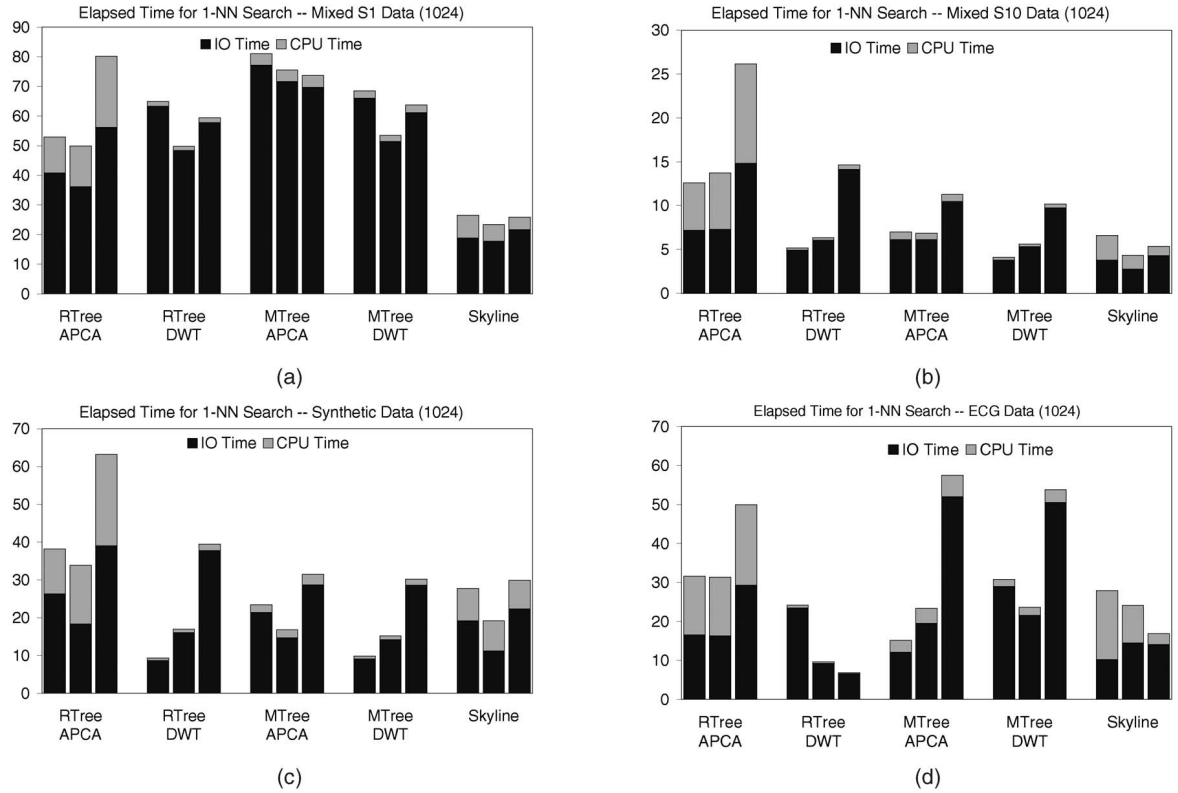


Fig. 7. Elapsed Time, in seconds, for 1-NN queries (for each method, three measurements from 16 (left), 32 (middle), and 64 (right) dimensional spaces). (a) Mixed S1 Data (1-NN, data length = 1,024). (b) Mixed S10 Data (1-NN, data length = 1,024). (c) Synthetic Data (1-NN, data length = 1,024). (d) ECG Data (1-NN, data length = 1,024).

objects fetched by each of the techniques evaluated in this study for k-NN queries (columns 2 to 7). This measure is free of implementation bias, allowing direct comparisons across all the evaluated techniques. These tables also show the *index search overhead* of each technique as the number of index pages accessed during search (columns 8 to 13). In Table 5, we show results for 1-NN queries on three data sets with time series of length 1,024. Table 6 shows results for the remaining data set for 10-NN queries and time series of length 256, 512, and 1,024.

Several observations can be made based on the results shown in Tables 5 and 6. First, as expected, there was a clear trade off between data and index IO depending on the dimensionality of the feature vector used to approximate time series. The number of data objects fetched decreased as we increased the dimensionality of the data approximation. The reason is that, with a higher dimensionality, the fidelity of approximation improves, thereby reducing the number of false alarms. On the other hand, increasing the dimensionality of the approximation had a negative effect on the index performance. As the dimensionality increased, the number of index pages accessed during search also increased.

Second, we also observed that the choice of index organizations had nontrivial impact on index search overhead. Consider, for example, the Haar wavelet transformation, where the data IO was identical for R-tree and M-tree indexes (due to the r-optimality of the k-NN algorithm, as discussed in Section 3.5). The impact of the index organization on the index search overhead is more evident for the ECG and Mixed S1 data sets, where the R-tree outperformed the M-tree by a wide margin. For the other two data sets, the R-tree still outperformed the M-tree when 16-dimensional feature vectors were used.

Third, if we compare the R-tree APCA and the Skyline indexes, we can observe the benefit of using skyline bounding regions on the index performance. Note that this is a valid comparison because both approaches are implemented on top of the R-tree interface provided by the GiST library. In both cases, the number of index pages accessed during k-NN queries increased as the dimensionality of the feature vectors increased. However, for the R-tree APCA, the number of index pages increased at a higher rate than for the Skyline index. This is consistent with our earlier observations on the quality of the bounding regions defined by the APCA index. Regions defined by the APCA index suffer from internal overlap. The Skyline index, on the other hand, defines more efficient skyline bounding regions free of internal overlap. Additionally, we observed performance gains on the number of data objects fetched by the Skyline index with respect to the R-tree APCA. This result was consistent with the theoretical basis defined in Section 3.5. These performance gains were more evident for the Mixed S1 and Mixed S10 data sets. Overall, the Skyline index yielded better IO performance in most of the experiments, except for only one case (ECG data set), where the R-tree DWT accessed a smaller number of index pages than the Skyline index.

4.4.2 Elapsed Time

In this section, we present the overall query performance as the elapsed time measured by wall-clock. We flushed the

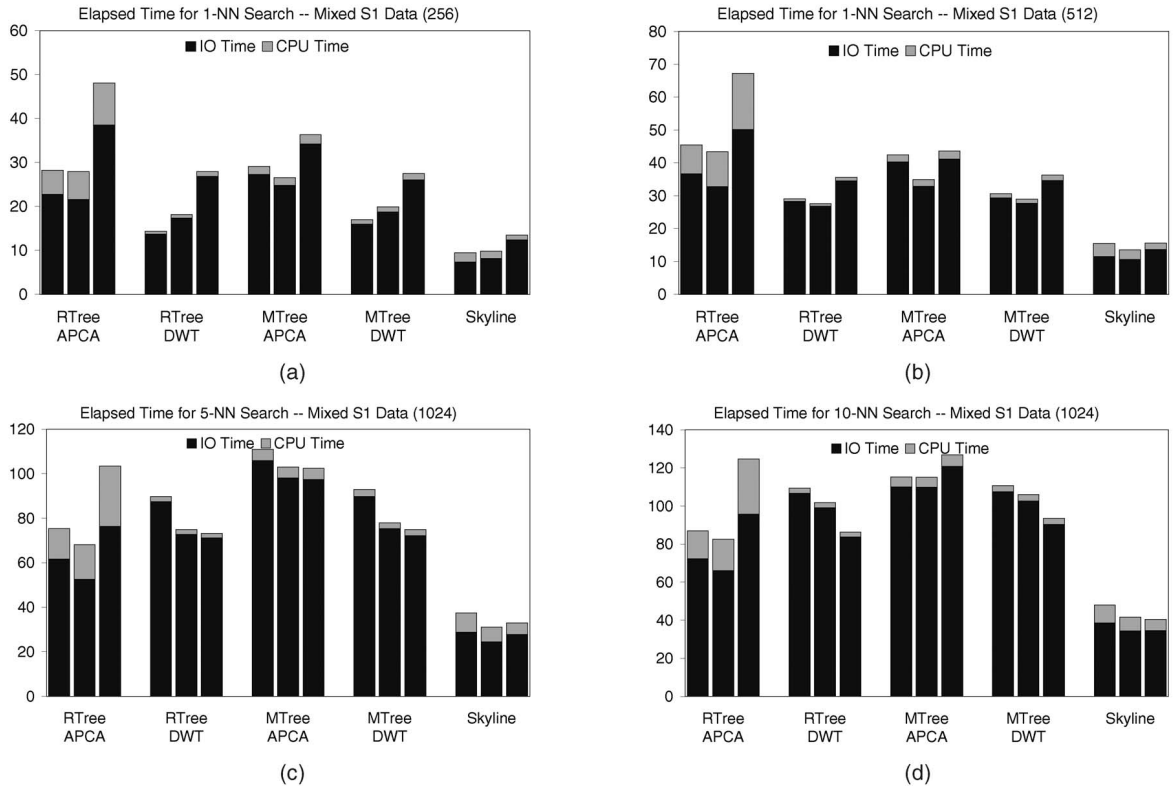


Fig. 8. Elapsed Time, in seconds, for k-NN queries (for each method, three measurements from 16 (left), 32 (middle), and 64 (right) dimensional spaces). (a) Mixed S1 Data (1-NN, data length = 256). (b) Mixed S1 Data (1-NN, data length = 512). (c) Mixed S1 Data (5-NN, data length = 1,024). (d) Mixed S1 Data (10-NN, data length = 1,024).

entire memory of the hardware system between consecutive runs of our experiments to avoid system caching effects. Furthermore, the portion of the CPU time was measured separately to profile the performance behaviors more accurately. We did not measure the elapsed time for *Hybrid-tree APCA* because the Hybrid-tree implementation we obtained from the authors of the APCA work [17] preloaded an entire index into memory before starting query processing and made it impossible to measure time spent on actual IO operations.

Fig. 7 shows the performance results of five different methods measured in elapsed time for 1-nearest neighbor queries. It was not surprising that the IO portion of the elapsed time closely followed the IO requirements shown in Tables 5 and 6. In general, the elapsed time spent on IO was the dominant factor of the overall execution time. In most of the cases, the Skyline index yielded the best overall performance. This was the result of the smallest IO requirements of the Skyline index both in number of data object fetched and in number of index pages accessed.

We also observed that *the CPU time cost was heavily affected by the choice of approximation methods.* With the same R-tree index, the APCA spent a significantly higher amount of CPU time than the Haar transformation (DWT). This was due to the different computational requirements to compute lower bound distance between a query and a time series data. Under the Haar transformation, query and data time series are in the same representation and its distance can be computed without further transformation. Under the APCA representation, on the other hand, the APCA representation of a query q has to be regenerated from its raw data every time a lower bound distance $d_{Apca}(q, x')$ needs to be computed. In fact, the Skyline index also suffers from the high computational cost since $d_{Apca}()$ is used. Nonetheless, the Skyline index still outperformed the other methods in most of the cases due to its substantially reduced IO cost.

The same trend was observed from time series data of length 256 and 512, as shown in Figs. 8a and 8b. Figs. 8c and 8d show the elapsed time for 5 and 10-nearest neighbor queries, respectively. With an increased number of nearest neighbors, the dominance of the IO time in the overall performance grew even larger. This trend suggests that the Skyline index is anticipated to outperform the other methods with wider margins for larger k values.

5 Discussion

Now that we have provided detailed descriptions of our contributions in this paper, we proceed to make a brief comparison with previous work. To the best of our knowledge, we are the first to use Skyline Bounding Regions (SBR) to organize and index time series data. An SBR is defined in the same *time-value* space where time series data are defined. Because of the large storage requirements of an SBR, the *Skyline index* only stores approximations of the SBR in the index nodes. Any approximation can be used to represent an SBR as long as the approximated SBR for an index node always bounds the time series data within the node. Previous work using approximations in the *time-value* space [18], [36], does not guarantee that the bounding region fully contains raw time series data. This is because the bounding regions are

generated only to enclose the approximation of time series data, but not the data itself. To present concrete definitions, we used the L_2 norm as a similarity metric in this paper. However, since our index structure does not transform the *time-value* space, any L_p norm could be used.

We did not directly address the subsequence match problem. This problem has been successfully studied and good techniques have been proposed for its solution [10], [15], [25], [26]. We believe that the *Skyline index* can be efficiently combined with these techniques for solving subsequence match queries.

6 Conclusion and Future Work

In this paper, we introduce the *Skyline Index*, a simple and elegant paradigm for indexing time series data. The Skyline index uses the Skyline Bounding Region (SBR) to group time series data. To lower bound the distance between a query time series and an SBR, the $LB_{Keogh}()$ function is used.

By analyzing the relationship between the feature distance of a time series data and the lower bounding distance of its bounding region from a query, we show that the *containment property* is a necessary condition for the r-optimality of multistep k-NN search algorithms. Without r-optimality, a multistep k-NN search algorithm might retrieve a different number of candidates depending on the choice of index organization. We also show that the *group lower bound property* is necessary to guarantee the correctness of k-NN search algorithms, but it cannot ensure retrieving the same set of candidates for different indexes, even if these indexes use the same feature representation.

Experimental results show that the Skyline Index can be coupled with the state-of-the-art dimensionality reduction techniques and can improve the performance of similarity search by up to a factor of 3.

We plan to further study other approximation techniques and the effects of different splitting algorithms for the Skyline Index construction. We also plan to apply the Skyline Index approach to different dimensionality reduction techniques (e.g., DWT and DFT) to further examine the applicability of this new indexing paradigm.

Acknowledgments

The authors would like to thank Eamonn Keogh and Kaushik Chakrabarti for sharing with them time series data sets and valuable comments about the APCA approach. They also want to thank Marco Patella for providing them with the most recent release of the M-tree code. This work was sponsored in part by the US National Science Foundation (NSF) CAREER Award (IIS-9876037), US NSF Grant No. IIS-0100436, and US NSF Research Infrastructure Program EIA-0080123. It was also supported by Consejo Nacional de Ciencia y Tecnología and Universidad Autónoma de Sinaloa, scholarship 117476. The authors assume all responsibility for the contents of the paper. The work of Inés Fernando Vega López was performed while he was doing doctorate studies at the Department of Computer Science, University of Arizona.

References

[1] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases," *Proc. Int'l Conf. Foundations of Data Organizations and Algorithms,* pp. 69-84, Oct. 1993.

[2] R. Agrawal, K.-I. Lin, H.S. Sawhney, and K. Shim, "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases," *Proc. 21st Very Large Databases (VLDB) Conf.,* pp. 490-501, Sept. 1995.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R^*-Tree: An Efficient and Robust Access Method for Points and Rectangles," *Proc. 1990 ACM-SIGMOD Conf.,* pp. 322-331, May 1990.

[4] D. Berndt and J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series," *Proc. Knowledge Discovery in Databases Workshop,* pp. 359-370, July 1994.

[5] K. Chakrabarti and S. Mehrotra, "The Hybrid Tree: An Index Structure for High Dimensional Feature Spaces," *Proc. 15th Int'l Conf. Data Eng.,* pp. 440-447, Mar. 1999.

[6] K.-P. Chan and A.W.-C. Fu, "Efficient Time Series Matching by Wavelets," *Proc. 15th Int'l Conf. Data Eng.,* pp. 126-133, Mar. 1999.

[7] P. Ciaccia, M. Patella, and P. Zezula, "M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces," *Proc. 23rd Very Large Databases (VLDB) Conf.,* pp. 426-435, Aug. 1997.

[8] C. Faloutsos, *Searching Multimedia Databases Content.* Boston: Kluwer Academic, 1996.

[9] C. Faloutsos, H. Jagadish, A. Mendelzon, and T. Milo, "A Signature Technique for Similarity-Based Queries," *Proc. Int'l Conf. Compression and Complexity of Sequences,* pp. 2-20, June 1997.

[10] C. Faloutsos, A.M. Ranganathan, and Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases," *Proc. 1994 ACM-SIGMOD Conf.,* pp. 419-429, May 1994.

[11] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," *Proc. 1984 ACM-SIGMOD Conf.,* pp. 47-57, June 1984.

[12] J.M. Hellerstein, J.F. Naughton, and A. Pfeffer, "Generalized Search Trees for Database Systems," *Proc. 21st Very Large Databases (VLDB) Conf.,* pp. 562-573, Sept. 1995.

[13] S. Hettich and S.D. Bay, The UCI KDD Archive, http://kdd.ics.uci.edu, 2002.

[14] K.J. Jacob and D. Shasha, FinTime—A Financial Time Series Benchmark, http://cs.nyu.edu/cs/faculty/shasha/fintime.html, Mar. 2000.

[15] T. Kahveci and A. Singh, "Variable Length Queries for Time Series Data," *Proc. 17th Int'l Conf. Data Eng.,* pp. 273-282, Apr. 2001.

[16] E. Keogh, "Exact Indexing of Dynamic Time Warping," *Proc. 28th Very Large Databases (VLDB) Conf.,* pp. 406-417, Aug. 2002.

[17] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani, "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases," *Proc. 2001 ACM-SIGMOD Conf.,* pp. 151-162, May 2001.

[18] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases," *Knowledge and Information Systems,* vol. 3, no. 3, pp. 263-286, 2000.

[19] E. Keogh and S. Kasetty, "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration," *Proc. Eighth ACM-SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 102-111, July 2002.

[20] E. Keogh and M. Pazzani, "Scaling up Dynamic Time Warping for Datamining Applications," *Proc. Sixth ACM-SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 285-289, Aug. 2000.

[21] E. Keogh and P. Smyth, "A Probabilistic Approach to Fast Pattern Matching in Time Series Databases," *Proc. Third ACM-SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 24-30, Aug. 1997.

[22] S.-W. Kim, S. Park, and W.W. Chu, "An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases," *Proc. 17th Int'l Conf. Data Eng.,* pp. 607-614, Apr. 2001.

[23] F. Korn, H.V. Jagadish, and C. Faloutsos, "Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences," *Proc. 1997 ACM-SIGMOD Conf.,* pp. 289-300, May 1997.

[24] G.B. Moody, MIT-BIH Database Distribution, http://ecg.mit.edu/index.html, 1999.

[25] Y.-S. Moon, K.-Y. Whang, and W.-S. Han, "GeneralMatch: A Subsequence Matching Method in Time-Series Databases Based on Generalized Windows," *Proc. 2002 ACM-SIGMOD Conf.,* pp. 382-393, June 2002.

[26] Y.-S. Moon, K.-Y. Whang, and W.-K. Loh, "Duality-Based Subsequence Matching in Time-Series Databases," *Proc. 17th Int'l Conf. Data Eng.*, pp. 263-272, Apr. 2001.
[27] S. Park, W.W. Chu, J. Yoon, and C. Hsu, "Efficient Searches for Similar Subsequences of Different Lengths in Sequence Databases," *Proc. 16th Int'l Conf. Data Eng.*, pp. 23-32, Feb.-Mar. 2000.
[28] 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 Int'l Conf. Data Eng.*, pp. 33-42, Feb.-Mar. 2000.
[29] I. Popivanov and R.J. Miller, "Similarity Search over Time-Series Data Using Wavelets," *Proc. 18th Int'l Conf. Data Eng.*, pp. 212-221, Feb.-Mar. 2000.
[30] D. Rafiei and A. Mendelzon, "Similarity-Based Queries for Time Series Data," *Proc. 1997 ACM-SIGMOD Conf.*, pp. 13-25, May 1997.
[31] D. Rafiei and A. Mendelzon, "Efficient Retrieval of Similar Time Sequences Using DFT," *Proc. Int'l Conf. Foundations of Data Organizations and Algorithms*, Nov. 1998.
[32] T. Seidl and H.-P. Kriegel, "Optimal MultiStep k-Nearest Neighbor Search, *Proc. 1998 ACM-SIGMOD Conf.*, pp. 154-165, May 1998.
[33] H. Shatkay and S. Zdonik, "Approximate Queries Representations for Large Data Sequences," *Proc. 12th Int'l Conf. Data Eng.*, pp. 536-545, Feb.-Mar. 1996.
[34] Z.R. Struzik and A.P.J.M. Siebes, "The Haar Wavelet Transform in the Time Series Similarity Paradigm," *Proc. Principles of Data Mining and Knowledge Discovery, Third European Conf.*, pp. 12-22, Sept. 1999.
[35] Y.-L. Wu, D. Agrawal, and A. El Abbadi, "A Comparison of DFT and DWT Based Similarity Search in Time-Series Databases," *Proc. Ninth ACM-CIKM Int'l Conf. Information and Knowledge Management*, pp. 488-495, Nov. 2000.
[36] B.-K. Yi and C. Faloutsos, "Fast Time Sequence Indexing for Arbitrary L_p Norms," *Proc. 26th Very Large Databases (VLDB) Conf.*, pp. 385-394, Sept. 2000.
[37] B.-K. Yi, H.V. Jagadish, and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences under Time Warping," *Proc. 14th Int'l Conf. Data Eng.*, pp. 201-208, Feb. 1998.

Quanzhong Li received the BS and MS degrees in computer science from Peking University in 1995 and in 1998. Since 1998, he has been at the University of Arizona, where he is a PhD student in the Computer Science Department. He is expected to obtain the PhD degree in August 2004. His research interests include all aspects of database systems, especially XML data processing, time series data, and multidimensional databases. He is also interested in Web technologies including high performance Web server systems, Web-based information discovery, and XML document processing.

Inés Fernando Vega López is a candidate for the PhD degree in computer science from the University of Arizona. He received the MS degree in computer science from the University of Arizona in 1999 and the BE degree in computer systems from the Instituto Tecnológico y de Estudios Superiores de Monterrey, Sinaloa México, in 1994. His current research interests include high-dimensional data indexing and query processing, multimedia, spatial, and temporal databases.

Bongki Moon received the PhD degree in computer science from the University of Maryland, College Park, in 1996, and the MS and BS degrees in computer engineering from Seoul National University, Korea, in 1985 and 1983, respectively. He is an associate professor of computer science at the University of Arizona. His current research areas of interest are XML indexing and query processing, high performance spatial and temporal databases, multidimensional databases, scalable web servers, and parallel and distributed processing. He was a member of the communication systems research staff of Samsung Electronics Corp. and the Samsung Advanced Institute of Technology, Korea, from 1985 to 1990. He received the US National Science Foundation CAREER Award in 1999.