LETTER

Physical Database Design for Efficient Time-Series Similarity Search

Sang-Wook KIM[†a)], **Jinho KIM**[††], ***Nonmembers*, *and*** **Sanghyun PARK**[†††], ***Member***

SUMMARY *Similarity search* in time-series databases finds such data sequences whose changing patterns are similar to that of a query sequence. For efficient processing, it normally employs a multi-dimensional index. In order to alleviate the well-known *dimensionality curse*, the previous methods for similarity search apply the *Discrete Fourier Transform (DFT)* to data sequences, and take only the first two or three DFT coefficients as *organizing attributes*. Other than this ad-hoc approach, there have been no research efforts on devising a systematic guideline for choosing the best organizing attributes. This paper first points out the problems occurring in the previous methods, and proposes a novel solution to construct optimal multi-dimensional indexes. The proposed method analyzes the characteristics of a target time-series database, and identifies the organizing attributes having the best *discrimination power*. It also determines the optimal number of organizing attributes for efficient similarity search by using a cost model. Through a series of experiments, we show that the proposed method outperforms the previous ones significantly.

key words: *time-series databases, similarity search, multi-dimensional indexes*

1. Introduction

A *time-series* database is a set of data sequences, each of which is an ordered list of elements. *Similarity search* is an operation that searches for the sequences or subsequences whose changing patterns are similar to that of a given query sequence [1], [2], [7]. For example, consider a time-series database that stores the stock history data. Similarity search is of growing importance in many new applications such as data mining and data warehousing [6].

Most approaches for similarity search regard a sequence with n elements as a point in n-dimensional space, and define the basic similarity of the two sequences by using the *Euclidean distance* between their corresponding points [1], [7], [10]. They also use a *multi-dimensional index* such as the R-tree, R^*-tree, and R^+-tree for efficient handling of multi-dimensional points. To avoid the *dimensionality curse* [1], [4], [8] in a multi-dimensional index, they usually apply the *Discrete Fourier Transform (DFT)* [9] to data sequences, and select the first two or three DFT coefficients as organizing attributes of the multi-dimensional index [1], [7].

Manuscript received August 7, 2006.
Manuscript revised October 29, 2007.

[†]The author is with the Division of Information and Communications, Hanyang University, 17 Haengdang-dong, Sungdong-gu, Seoul, 133-791, Korea.
[††]The author is with the Department of Computer Science, Kangwon National University, Korea.
[†††]The author is with the Department of Computer Science, Yonsei University, Korea.
a) E-mail: wook@hanyang.ac.kr

DOI: 10.1093/ietcom/e91–b.4.1251

However, there have been no research efforts to devise a systematic approach for selecting the best organizing attributes from a set of DFT coefficients. This paper mainly focuses on this issue. First, we point out the performance problem of the previous methods, and then propose a novel method for constructing a multi-dimensional index as its solution. The proposed method analyzes the characteristics of a target time-series database, and then selects the organizing attributes of the multi-dimensional index with a high discrimination power. It also determines the optimal number of organizing attributes for efficient similarity search using the proposed cost model. We compare the proposed method with the previous ones to verify its effectiveness.

2. Previous Work

2.1 Similarity Search

The similarity measure of any two sequences, $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n)$, widely-used in a time-series database, is the *Euclidean distance* $D(X, Y)$ [1], [7] defined as follows:

$$D(X, Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

Similarity search is defined as the operation that finds such data sequences Y whose Euclidean distances to a query sequence X are within a given distance tolerance ε.

2.2 Index Construction

Typically, previous methods [1], [2], [7], [10] regard sequences with n elements as points in n-dimensional space, and utilize a multi-dimensional index for indexing them. The storage and computation costs of a multi-dimensional index grow exponentially with n [4], [8]; The multi-dimensional index suffers from serious performance degradation when n is large, which is general in a time-series database [1], [2]. We call it *dimensionality curse*.

To overcome such a problem, the previous work usually depends on dimensionality reduction techniques using the *Discrete Fourier Transform (DFT)* and the *Discrete Cosine Transform (DCT)* [9]. The DFT converts a sequence $X = (x_1, x_2, ..., x_n)$ of *real-valued* elements in time domain into a sequence $X_F = (x_{F_1}, x_{F_2}, ..., x_{F_n})$ of *complex-valued* elements in frequency domain. Each complex-valued element x_{F_k} ($k = 1, 2, ...n$) in frequency domain is defined as follows:

$$x_{F_k} = \frac{1}{\sqrt{n}} \sum_{t=1}^{n} x_t \exp\left(\frac{-j2\pi kt}{n}\right) \qquad (2)$$

where j is the imaginary unit ($= \sqrt{-1}$).

The Parseval's theorem proves that the *energy* of a sequence in time domain is always identical to that of the same sequence in frequency domain. Therefore, the DFT preserves the features of a sequence after the conversion. In addition, the first few DFT coefficients contain a majority of the energy of the sequence in frequency domain.

Using these two properties, the previous methods [1], [2], [7] take the first k ($k \ll n$) DFT coefficients from the sequences after conversion, and construct a multi-dimensional index with $2 \times k$ coefficients as organizing attributes. Since k is much smaller than n, the storage and computation costs of a multi-dimensional index reduce significantly. However, they introduce the *false match* [1] because a candidate set obtained from index searching contains such sequences whose actual distances to a query sequence are not within the distance tolerance ε. This is mainly due to the *energy loss* caused by the DFT coefficients excluded for organizing attributes. Therefore, they make a candidate set go through the final validation step for detecting and discarding false matches.

2.3 Problems

The cost of processing similarity search consists of two parts: the index access cost for index searching and the object access cost for resolving false matches. In case of large k, similarity search requires a small object access cost thanks to little energy loss, but a large index access cost due to the problem of the dimensionality curse. In case of small k, on the contrary, the index access cost becomes smaller while the object access cost gets larger. Therefore, it is desirable to devise a systematic method that constructs an optimal multi-dimensional index that minimizes both the object access and index access costs simultaneously. The previous methods have the following problems in the sense of optimality in constructing their index structures.

1. They simply consider the *energy level* of the DFT coefficients as criteria in choosing organizing attributes. We know that the organizing attributes should have a discrimination power higher than others. A high energy level, however, does not always imply a high discrimination power. For example, it is worthless to choose a DFT coefficient having the highest level of energy if all the sequences in a database have the same level of energy for this DFT coefficient.
2. They do not provide a systematic guideline for determining the optimal number of organizing attributes. The previous methods just recommend the first *two or three* DFT coefficients. However, we expect that the optimal number of organizing attributes would be dependent on the characteristics of a target database.
3. They take both of the real and imaginary parts of a DFT coefficient together as organizing attributes in index construction. However, the real and the imaginary parts have different discrimination powers even though they come from the same DFT coefficient. Therefore, it is better to treat them independently.

3. Proposed Method

3.1 Basic Idea

We propose to use the following strategies to solve the problems aforementioned.

1. The real and imaginary parts are treated separately even though they are from the same DFT coefficient. For this separation, the new terminology, the *element coefficient* is defined as the real *or* the imaginary part of a DFT coefficient. Thus, the task of choosing organizing attributes is performed in the unit of element coefficients. This means that real and imaginary parts of a DFT coefficient are independently evaluated in the process of choosing organizing attributes. As a result, it is possible that while the real part of a DFT coefficient has been chosen, its imaginary part has not been chosen as organizing attributes. It is also possible that the real part of a DFT coefficient has not been chosen while its imaginary part has been chosen as organizing attributes.
2. The *standard deviation* of the element coefficient defined below is used as criteria in choosing organizing attributes.

$$Std_{ec_i} = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(ec_i(j) - mean_{ec_i})^2},$$
$$mean_{ec_i} = \frac{1}{N}\sum_{j=1}^{N} ec_i(j),$$
$$\textit{where } ec_i(j) \textit{ denotes the value of the } i\text{th } \textit{element coefficient for the } j\text{th } \textit{sequence.} \qquad (3)$$

A large standard deviation of an element coefficient implies that a large number of sequences in a database have different levels of energy for that element coefficient. Thus, we say that the larger the standard deviation, the higher the discrimination power. In this sense, the element coefficients to be included in organizing attributes are those with large standard deviations.
3. We analyze the characteristics of a target database in advance using a cost model. Through this analysis, we estimate the index and object access costs, which are used subsequently to determine the optimal number of organizing attributes.

3.2 Cost Model

Similarity search requires both index and object accesses. Because the cost of disk accesses typically dominates in a time-series database environment, we develop a cost model that mainly considers the number of disk accesses. Assuming that the R*-tree, the most popular multi-dimensional index at present, is employed for indexing, our cost model combines and extends the previous ones [3], [5] that use the *fractal dimension* D_0 and the *correlation fractal dimension* D_2.

Our cost model is based on the following equation.

$$D_q = \frac{1}{q-1} \frac{\delta \log \sum_i (p_i)^q}{\delta \log r} \tag{4}$$

It becomes the fractal dimension when $q = 0$ and the correlation fractal dimension when $q = 2$. Here, p_i is the number of points inside the i^{th} cell when we divide the multi-dimensional space into the hyper-cubic grid cells of side r. In reality, these values are obtained by the algorithms presented in [3].

3.2.1 Index Access Cost

The index access cost is estimated by the following expression proposed in [5].

$$IndexAccessCost = \sum_{j=0}^{h-1} \frac{N}{C_{eff}{}^{h-j}} \prod_{i=0}^{k} (\sigma_j + \varepsilon) \tag{5}$$

Here, N is the total number of data sequences, and k is the number of organizing attributes participating in the R*-tree. C_{eff}, the average number of entries per node, represents the effective capacity of a node of the R*-tree. h is the height of the R*-tree (the root is assumed at level $j = 0$ and the leaves at level $j = h-1$). ε is the distance tolerance given with a query sequence, and σ_j is the side of the square minimum bounding rectangle (MBR) of a node at level j. Based on the fractal dimension D_0, σ_j is actually computed by the expression:

$$\sigma_j = \left(\frac{C_{eff}{}^{h-j}}{N} \right)^{\frac{1}{D_0}} \tag{6}$$

3.2.2 Object Access Cost

The object access cost is estimated by the expression, a slightly modified version of the one proposed in [3].

$$ObjectAccessCost = \left(\frac{Vol(\varepsilon + \alpha, \odot)}{Vol(\varepsilon + \alpha, \square)} \right)^{\frac{D_2}{n}} \times (N-1) \times 2^{D_2} \times (\varepsilon + \alpha)^{D_2} \tag{7}$$

Here, D_2 is the correlation fractal dimension, n is the number of elements in a sequence, and ε is the distance tolerance. α is the sum of standard deviations of element coefficients not chosen as organizing attributes. $Vol(\varepsilon+\alpha, \square)$ is the volume of the hyper-cubic cell of side $\varepsilon+\alpha$, and $Vol(\varepsilon+\alpha, \odot)$ the volume of the hyper-sphere with radius $\varepsilon + \alpha$. N is the number of data sequences in a database.

Algorithm 1

```
INPUT: Set of $N$ data sequences with $n$ elements, set of $q$
       query sequences, and a distance tolerance $\varepsilon$ frequently used in
       queries

OUTPUT: Set of element coefficients selected as organizing
        attributes for an R*-tree

1. Compute the fractal dimension $D_0$ and the correlation fractal
   dimension $D_2$ by analyzing all the data sequences stored in a
   database

2. Access every data sequence from the database and perform the
   DFT

3. Sort the element coefficients in the descending order of their stan-
   dard deviation, and insert the identifiers of the element coeffi-
   cients into the array $EC(i)$ $(1 \le i \le 2n)$ in the sorted order

4. For each $i$ do
     a. Suppose that an R*-tree is built by using $i$ organizing at-
        tributes $EC(1), EC(2), ..., EC(i)$
     b. Given each query sequence $Q(j)$ $(1 \le j \le q)$ and $\varepsilon$, com-
        pute the index access cost $IndexAccessCost(i, j)$ and the
        object access cost $ObjectAccessCost(i, j)$ when utilizing
        the multi-dimensional index supposed in the line 5
     c. Using $IndexAccessCost(i, j)$ and
        $ObjectAccessCost(i, j)(1 \le j \le q)$, estimate $TotalCost(i)$,
        the total cost (i.e., index access cost+object access cost) for
        processing all the $q$ queries using the R*-tree with $i$ orga-
        nizing attributes

5. Find $TotalCost(w)$, the smallest one among $TotalCost(i)$
   $(1 \le i \le 2n)$

6. Return element coefficients $EC(1), EC(2), ..., EC(w)$ as the orga-
   nizing attributes of the R*-tree
```

3.3 Algorithm

The detailed procedure for R*-tree construction is shown in Algorithm 1.

4. Performance Evaluation

4.1 Environment for Experiments

The data and query sequences were generated using the expression used in [1]. First, using the expression $x_i = x_{i-1} + rand()$, we generated 10,000 data sequences $X = (x_i)$. Here, $rand()$ is the random function that picks the data value uniformly from the range of $[-500, 500]$. Every sequence has 32 elements ($n = 32$). To generate 1,000 query sequences $Q = (q_i)$, we used the expression $q_i = x_i + 0.05 * rand()$. Using $\sqrt{1000 \times n}$ as the base value ε_1 of a distance tolerance, we set ε_i as $\varepsilon_1 \times i$. The page size of the R*-tree and data was

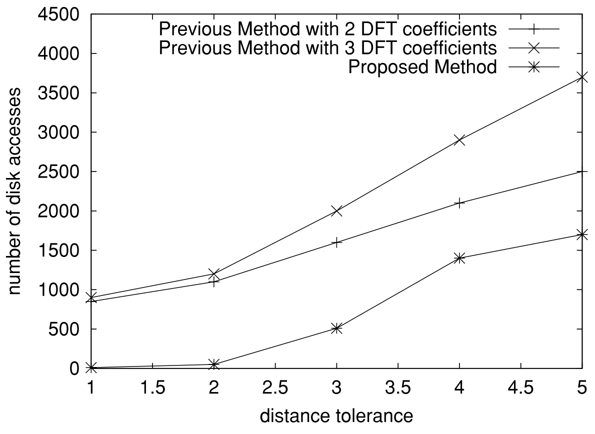

Fig. 1 Performance comparisons of the proposed and previous methods in similarity search.

all set to 512 bytes. Every sequence was stored in a separate page to nullify the clustering effect.

4.2 Experimental Result

Figure 1 shows the experimental results. The X-axis is the distance tolerance submitted with a query sequence, and the Y-axis is the average number of disk accesses (object accesses+index accesses), which actually occurred during processing 1,000 query sequences. For indexing, previous methods took the first two or three DFT coefficients, respectively, as organizing attributes while our method used the element coefficients chosen by Algorithm 1.

The figure shows that (1) the performance improvement of our method is significant (from 2 times to 100 times), and (2) the performance benefit gets larger when the distance tolerance becomes small. This is quit desirable since users are normally interested in just a few number of final answers in actual situations. We also performed the experiments using different sets of data and query sequences but the results appeared similar. Therefore, the experimental results verified that the proposed method improves the performance of similarity search greatly.

5. Conclusions

A multi-dimensional index is essentially used for efficient similarity search in time-series databases. Since searching in high-dimensional space suffers from the dimensionality curse problem, most previous methods employ the dimensionality reduction techniques such as the DFT to convert the sequences into the points on low-dimensional space. However, there has been no discussion on the systematic way that guides 1) which coefficients are to be chosen, and 2) how many coefficients are to be selected in constructing the index for efficient query processing.

This paper dealt with this issue and proposed a unique method that enables to construct an R*-tree for efficient similarity search. First, it regards the real and imaginary parts of a complex DFT coefficient separately in order to prevent the element coefficients with a low discrimination power from being chosen as organizing attributes. Second, it selects the element coefficients with a high discrimination power. As criteria for evaluating a discrimination power, it uses the standard deviation. Third, using the cost model appropriate for similarity search, it estimates the costs of the index and object accesses, and determines the number of organizing attributes systematically.

For choosing good organizing attributes, our method requires an additional analysis process, where it reads all the sequences for computing the standard deviation of every element coefficient. Thus, it requires an extra cost of accessing an entire time-series database from disk. However, we note that this analysis process is performed off-line, thus does not affect the performance of on-line query processing. The experimental results show that the proposed method outperforms the previous ones considerably.

Acknowledgement

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2007-C1090-0701-0040).

References

[1] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," Proc. Int'l. Conf. on Foundations of Data Organization and Algorithms (FODO), pp.69–84, Oct. 1993.

[2] S.H. Lim, H.J. Park, and S.W. Kim, "Using multiple indexes for efficient subsequence matching in time-series databases," Proc. Int'l. Conf. on Database Systems for Advanced Applications (DASFAA), pp.65–79, April 2006.

[3] A. Belussi and C. Faloutsos, "Estimating the selectivity of spatial queries using the correlation fractal dimension," Proc. Int'l. Conf. on Very Large Data Bases (VLDB), pp.299–310, Sept. 1995.

[4] S. Berchtold, D.A. Keim, and H.-P. Kriegel, "The X-tree: An index structure for high-dimensional data," Proc. Int'l. Conf. on Very Large Data Bases (VLDB), pp.28–39, 1996.

[5] C. Faloutsos and I. Kamel, "Beyond uniformity and independence: Analysis of R-trees using the concept of fractal dimension," Proc. ACM Int'l. Conf. on Management of Data (ACM SIGMOD), pp.4–13, May 1994.

[6] S.W. Kim, S.H. Park, and W.W. Chu, "Efficient processing of similarity search under time warping in sequence databases: An index-based approach," Information Systems, vol.29, no.5, pp.405–420, July 2004.

[7] W.K. Loh, S.W. Kim, and K.Y. Whang, "A subsequence matching algorithm that supports normalization transform in time-series databases," Data Mining and Knowledge Discovery Journal, vol.9, no.1, pp.5–28, July 2004.

[8] S.W. Kim, C. Aggarwal, and P.S. Yu, "Effective nearest neighbor indexing with the euclidean metric," Proc. ACM Int'l. Conf. on Information and Knowledge Management (ACM CIKM), pp.9–16, 2001.

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

[10] Y. Moon, K. Whang, and W. Han, "A subsequence matching method in time-series databases based on generalized windows," Proc. ACM Int'l. Conf. on Management of Data (ACM SIGMOD), pp.382–393, June 2002.