Proc. 22nd IEEE Int. Conf. on Data Engineering (ICDE'06), Atlanta, GA, 2006

Threshold Similarity Queries in Large Time Series Databases

Johannes Aßfalg, Hans-Peter Kriegel, Peer Kröger, Peter Kunath, Alexey Pryakhin, Matthias Renz
Institute for Computer Science, University of Munich
{assfalg,kriegel,kroegerp,kunath,pryakhin,renz}@dbs.ifi.lmu.de

Abstract

Similarity search in time series data is an active area of research. In this paper, we introduce the novel concept of threshold-similarity queries in time series databases which report those time series exceeding a user-defined query threshold at similar time frames compared to the query time series. In addition, we present a new data structure to support threshold similarity queries efficiently. The performance of our solution is demonstrated by an extensive experimental evaluation.

1 Introduction

Similarity search in time series data has attracted a lot of research work recently. In this paper, we introduce a novel type of similarity queries on time series databases called *threshold similarity queries*. A threshold similarity query is defined by a query time series Q and a threshold τ. The database time series as well as the query sequence Q are decomposed into time intervals of subsequent elements where the values are (strictly) above τ. Now, the threshold similarity query returns those time series which have a similar interval sequence of values above τ. Note, that the entire set of absolute values are irrelevant for the query as long as they exceed the threshold τ.

The novel concept of threshold similarity queries is an important technique useful for many practical application areas. In pharmaceutical industry it can help to identify drugs that cause similar effects in the blood values of a patient at the same time after the drug application. Obviously, effects like a certain blood parameter exceeding a critical level τ are of particular interest. For environment observation applications, a topic of research is the detection of dependencies between different air pollution attributes, e.g. the detection of attributes which nearly simultaneously exceed their legal threshold. Queries like "return all ozone time series which exceed the threshold $\tau_1 = 50\mu g/m^3$ at a similar time as the temperature reaches the threshold $\tau_2 = 25°C$" require an efficient support of threshold similarity queries. In molecular biology the analysis of gene expression data is important to understand cellular mechanisms. Biologists search for genes that have a similar up and down pattern of their expression level over time because this indicates a functional relationship among the particular genes. Since the absolute up/down-value is irrelevant, this problem can be represented by a threshold similarity query with a threshold of $\tau = 0$.

In this paper, we make the following contributions. We formalize the novel concept of threshold similarity queries on time series databases. In addition, we present a novel data representation of time series which supports such threshold similarity queries efficiently. Finally, we present an experimental evaluation that includes performance tests of our proposed algorithms and shows that our new concept of threshold queries can be successfully employed in several application fields.

The remainder is organized as follows. We briefly overview related work in Section 2. Section 3 formalizes the concept of threshold similarity queries. In Section 4, we show how time series can be represented in order to support threshold similarity queries efficiently. The effectiveness and efficiency of our algorithm are evaluated in Section 5. Section 6 concludes the paper.

2 Related Work

A lot of work on similarity search in time series databases has been published recently. The proposed methods mainly differ in the representation of the time series, a survey is given in [3]. However, all proposed approaches usually cannot be applied to our novel problem of threshold similarity queries. For example, techniques which are based on dimension reduction suffer from the problem that temporal information is lost. Usually, in a reduced feature space, the original inter-

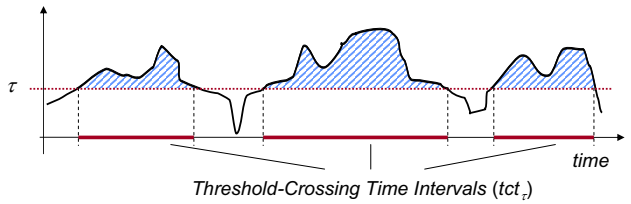


Figure 1. Threshold-Crossing Time Intervals

vals indicating that the time series is above a given threshold cannot be generated. Specialized distance functions, e.g. dynamic time warping (DTW) [1] that considers the absolute values of the time series rather than the intervals of values above a given threshold are also not applicable to threshold similarity queries. In [4], a novel bit level approximation of time series for similarity search is proposed. Each value of the time series is approximated by a bit which is set to 1 if the value is strictly above the mean value of the entire time series, otherwise it is set to 0. A distance function is defined on this bit level representation that lower bounds the Euclidean distance and, by using a variant, lower bounds DTW, too. However, since this representation is restricted to a certain predetermined threshold, this approach is also not applicable for threshold queries where the threshold is not known until query time.

3 Threshold Similarity Queries on Time Series

A time series X is a sequence of values $x_i \in \mathbb{R}$ $(i = 1 \ldots N)$ at different points $t_i \in T$ in time, where T denotes the domain of time and $\forall i \in \{1, .., N-1\}$: $t_i < t_{i+1}$. Let us note that we assume that missing continuous values are linearly interpolated from discrete measurements. Then, a *threshold-crossing time interval sequence* of a time series $X = \langle x_i \in \mathbb{R} : i = 1..N \rangle$ w.r.t. a threshold $\tau \in \mathbb{R}$ denoted by $TCT_\tau(X)$ is the smallest sequence $TCT_\tau(X) = \langle (l_j, u_j) \in T \times T : j \in \{1, .., M\}, M \leq N \rangle$ of time intervals, such that

$$\forall t \in T : (\exists j \in \{1, .., M\} : l_j < t < u_j) \Leftrightarrow x_t > \tau.$$

An interval $tct_{\tau,j} = (l_j, u_j)$ of $TCT_\tau(X)$ is called *threshold-crossing time interval*. This concept is visualized in Figure 1.

In the following, we consider time intervals as points in a two dimensional space (*time interval plane*). This plane is spanned by the starting times (first dimension) and the ending times (second dimension) of intervals. Consequently a threshold-crossing time interval sequence is represented by a set of 2-dimensional points in the time interval plane.

We define two time intervals to be similar if they have similar starting and ending points. In the time interval plane this similarity inversely corresponds to the Euclidean distance of the associated points. To compute the similarity $d_{TS}(X, Y, \tau)$ of two threshold-crossing time interval sequences X and Y for a threshold τ, we use the Sum of Minimum Distances (SMD)[2] as it adequately reflects the notion of similarity between two point sets in the time interval plane.

Based on the similarity function defined on threshold-crossing time interval sequences, we define the *Threshold Similarity Query* as follows: For a given parameter k, a query time series Q, and a threshold τ, the Threshold Similarity Query yields the k-Nearest-Neighbors of Q with respect to the similarity of the corresponding threshold-crossing time interval sequences. Note that we set $k = 1$ if not stated otherwise.

4 Efficient Management of Threshold-Crossing Time Intervals

The simplest way to execute a threshold similarity query is to sequentially read each time series X from the database, to compute the threshold-crossing time interval sequence $TCT_\tau(X)$ and to compute the threshold-similarity function $d_{TS}(X, Y, \tau)$. Finally, we report this time series which yield the smallest $d_{TS}(X, Y, \tau)$. However, if the time series database contains a large number of objects and the time series are reasonably large, then obviously this type of performing the query becomes unacceptably expensive.

The basic idea of our approach is to pre-compute the $TCT_\tau(X)$ for all threshold values for each time series object X and store it on disk in such a way it can be accessed efficiently. Due to this pre-computation we do not need to access the complete time series data at query time. Instead only partial information of the time series objects is required to execute the query, which saves a lot of I/O cost.

4.1 Trapezoid Decomposition of Time Series

The set of all time intervals which start and end at the same time series segment can be described by a single trapezoid whose left and right bounds are each congruent with one single time series segment. Let $s_l = ((t_{l1}, x_{t_{l1}}), (t_{l2}, x_{t_{l2}}))$ denote the segment of the left bound and $s_r = ((t_{r1}, x_{t_{r1}}), (t_{r2}, x_{t_{r2}}))$ denote the segment of the right bound. The top-bottom bounds correspond to the two threshold-crossing time intervals $tct_{\tau_{top}}$ and $tct_{\tau_{bottom}}$ whose threshold values are computed as follows:

$$\tau_{top} = min(max(x_{t_{l1}}, x_{t_{l2}}), max(x_{t_{r1}}, x_{t_{r2}}))$$
$$\tau_{bottom} = max(min(x_{t_{l1}}, x_{t_{l2}}), min(x_{t_{r1}}, x_{t_{r2}}))$$

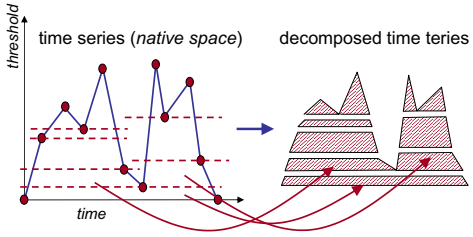


Figure 2. Time Series Decomposition

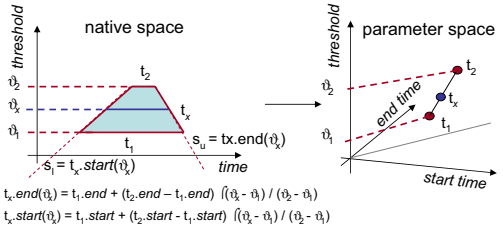


Figure 3. Interval Ranges in Parameter Space

For our decomposition algorithm we can use the following property. Threshold-crossing time intervals always start at increasing time series segments (positive segment slope) and end at decreasing time series segments (negative segment slope). Obviously, all values of X within the threshold-crossing time interval $tct_\tau(X)$ are greater than the corresponding threshold value τ. Let us assume that the time series segment s_l which lower-bounds the time interval at time t_l has a negative slope. Then all x_t on s_l with $t > t_l$ are lower than τ which contradicts the definition of threshold-crossing time intervals. The property of the ending segment can be made clear analogously.

Based on this observation, we developed an algorithm for the decomposition of the time series into corresponding trapezoids (cf. Figure 2) in linear time w.r.t. the length of the time series.

4.2 Indexing Segments of the Parameter Space

The threshold similarity of time series is computed in the time interval plane for a certain threshold. In order to support threshold similarity queries for arbitrary thresholds, we transform the trapezoids into segments in a three-dimensional space which we call *parameter space*. This space is spanned by the time interval plane and an additional dimension for the threshold values. An example is depicted in Figure 3. We apply the R*-tree for the efficient management of the three-dimensional segments representing the time series objects in the parameter space. As the R*-tree index can only manage rectangles, we represent the three-dimensional segments by rectangles where the segments correspond to one of the diagonals of the rectangles.

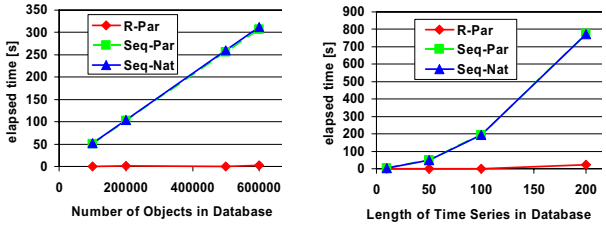
In fact, for all trapezoids which result from the time series decomposition, the lower bound time interval covers the upper bound time interval. Furthermore, intervals which are covered by another interval are located in the lower-right area of this interval representation in the time interval plane, as depicted in Figure 3. Consequently, the locations of the segments within the rectangles in the parameter space are fixed. Therefore, in the parameter space the bounds of the rectangle which represents a segment suffice to uniquely identify the represented segment. Let $((x_l, y_l, z_l), (x_u, y_u, z_u))$ be the coordinates of a rectangle in the parameter space. Then the coordinates of the corresponding segment are $((x_l, y_u, z_l), (x_u, y_l, z_u))$.

5 Experimental Evaluation

We compared the efficiency of our proposed approach (in the following denoted by 'R_{Par}') for answering threshold similarity queries using one of the following techniques: The first competitor, denoted by 'Seq_{Nat}', works with the native time series. At query time the threshold-crossing time intervals (TCT) are computed for the query threshold and afterwards the distance between the query time series and each database object can be derived. The second competitor, denoted by 'Seq_{Par}', works on the parameter space rather than on the native data. It stores all TCTs without using any index structures, i.e. a sequential scan over the elements of the parameter space is performed for query evaluation. All experiments were performed on a workstation featuring a 1.8 GHz Opteron CPU and 8GB RAM. We used a disk with a transfer rate of 100 MB/s, a seek time of 3 ms and a latency delay of 2 ms. Performance is presented in terms of the elapsed time including I/O and CPU-time.

We used several synthetic datasets and two real-world data sets for our evaluation. The real-world data sets are derived from two different applications: the analysis of environmental air pollution and gene expression data analysis. The data on environmental air pollution is derived from the Bavarian State Office for Environmental Protection, Augsburg, Germany [1] and contains the daily measurements of 8 sensor stations distributed in and around the city of Munich from the year 2000 to 2004. Each time series represents the measurement of one station at a given day containing 48 values for one of 10 different parameters such as temperature, ozone concentration, etc. The data on gene expression from [5] contains the expression level of approximately 6,000 genes measured at 24 different time slots.

[1] www.bayern.de/lfu



(a) Scalability w.r.t. database size.

(b) Scalability w.r.t. time series length.

Figure 4. Performance Results

5.1 Performance Results

At first we performed threshold similarity queries against databases of different sizes to measure the influence of the database size. The elements of the databases are time series of fixed length l. To obtain more reliable and significant results we used 5 randomly chosen query objects. Furthermore, these query objects were used in conjunction with 5 different thresholds. We obtained 25 different threshold similarity queries. Figure 4(a) exhibits the performance results for each database averaged over the 25 queries. Second, we explored the impact of the length of the query object and the time series in the database. We randomly chose 5 query time series objects and combined them with appropriate thresholds. This yielded 25 threshold similarity queries that were executed on the databases containing time series of different length. The results are shown in Figure 4(b). In both experiments our technique outperforms the competing approaches whose cost increase very fast due to the expensive distance computations. The results show that our approach scales very well even for large databases and is hardly influenced by the size of the time series objects.

5.2 Results on Real-World Datasets

We performed 10-nearest neighbor threshold queries with randomly chosen query objects on the air pollution dataset. Interestingly, when we choose time series as query objects, that were derived from rural sensor stations representing particulate matter parameters (M_{10}), we obtained only time series also measured at rural stations. This confirms that our novel query type is able to detect the differences between rural and urban pollution measurements.

The results on the gene expression dataset were also very interesting. The task was to find the most similar gene w.r.t. $\tau = 0$ to a given query gene. We posed several randomized queries to this dataset with $\tau = 0$ and evaluated the results w.r.t. biological interestingness using the SGD database [2]. Indeed, we retrieved functionally related genes for most of the query genes. For example, for query gene CDC25 we obtained the gene CIK3. Both genes play a role during the mitotic cell cycle.

To sum up, the results on the real-world datasets suggest the practical relevance of threshold queries for important real-world applications.

6 Conclusions

In this paper, we proposed a novel type of query on time series databases called threshold similarity query. Given a query object Q and a threshold τ, a threshold similarity query returns those time series in the database that exhibit the most similar threshold-crossing time interval sequence. The threshold-crossing time interval sequence of a time series represents the interval sequence of elements that have a value above the threshold τ. We presented a novel approach for managing time series data to efficiently support such threshold similarity queries. Our experimental evaluation demonstrates the importance of the new query type and shows the scalability of our proposed approach.

References

[1] D. Berndt and J. Clifford. "Using dynamic time warping to find patterns in time series". In *AAAI-94 Workshop on Knowledge Discovery in Databases*, 1994.

[2] T. Eiter and H. Mannila. "Distance Measure for Point Sets and Their Computation". In *Acta Informatica, 34*, pages 103–133, 1997.

[3] E. Keogh, K. Chakrabati, S. Mehrotra, and M. Pazzani. "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'01), Santa Barbara, CA*, 2001.

[4] C. A. Ratanamahatana, E. Keogh, A. J. Bagnall, and S. Lonardi. "A Novel Bit Level Time Series Representation with Implication for Similarity Search and Clustering". In *Proc. 9th Pacific-Asian Int. Conf. on Knowledge Discovery and Data Mining (PAKDD'05), Hanoi, Vietnam*, 2005.

[5] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, and B. Futcher. "Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization.". *Molecular Biolology of the Cell*, 9:3273–3297, 1998.

[2] http://www.yeastgenome.org/