In Proc. 4th Asia Pacific Bioinformatics Conference (APBC'06), Taipei, Taiwan, 2006.

SEMI-SUPERVISED THRESHOLD QUERIES ON PHARMACOGENOMICS TIME SEQUENCES

J. ASSFALG, H.-P. KRIEGEL, P. KRÖGER, P. KUNATH, A. PRYAKHIN, M. RENZ

Institute for Computer Science, University of Munich
Email: {assfalg,kriegel,kroegerp,kunath,pryakhin,renz}@dbs.ifi.lmu.de

The analysis of time series data is of capital importance for pharmacogenomics since the experimental evaluations are usually based on observations of time dependent reactions or behaviors of organisms. Thus, data mining in time series databases is an important instrument towards understanding the effects of drugs on individuals. However, the complex nature of time series poses a big challenge for effective and efficient data mining. In this paper, we focus on the detection of temporal dependencies between different time series: we introduce the novel analysis concept of *threshold queries* and its semi-supervised extension which supports the parameter setting by applying training datasets. Basically, threshold queries report those time series exceeding an user-defined query threshold at certain time frames. For semi-supervised threshold queries the corresponding threshold is automatically adjusted to the characteristics of the data set, the training dataset, respectively. In order to support threshold queries efficiently, we present a new efficient access method which uses the fact that only partial information of the time series is required at query time. In an extensive experimental evaluation we demonstrate the performance of our solution and show that semi-supervised threshold queries applied to gene expression data are very worthwhile.

1. Introduction

Data mining in time series data is a key step within the study of drugs and their impact on living systems, including the discovery, design, usage, modes of action, and metabolism of chemically defined therapeutics and toxic agents. In particular, the analysis of time series data is of great practical importance for pharmacogenomics.

Classical time series analysis is based on techniques for forecasting or for identifying patterns (e.g. trend analysis or seasonality). The similarity between time series, e.g. similar movements of time series, plays a key role for the analysis. In this paper, we introduce a novel but very important similarity query type which we call *threshold query*. Given a time series database DB, a query time series Q, and a query threshold τ, a threshold query $TSQ_{DB}(Q, \tau)$ returns those time series $X \in DB$ having the most similar sequence of time intervals in which the time series values are above τ. In other words, we assume that each time series $X \in DB \cup \{Q\}$ is transformed into a sequence of disjoint time intervals covering only those values of X that are (strictly) above the threshold τ. Then, a threshold query returns for a given query object Q that object $X \in DB$ having the most similar sequence of time intervals. Let us note that the exact values of the time series are not considered, rather we are only interested in whether the time series is above or below a given threshold τ. In other words, the concept of threshold queries enables us to focus only on the duration of certain events indicated by increased time series amplitudes, while the degree of the corresponding amplitudes are ignored. This advantage is very beneficial, in particular, if we want to compare time

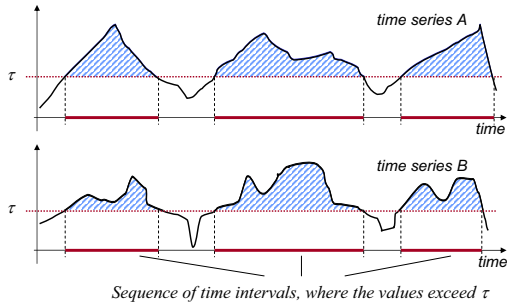


Figure 1. Illustration of transformation of time series into sequences of time intervals.

series reacting on certain stimulations with different sensitivity. The transformation of the time series into interval sequences is visualized in Figure 1. Two time series A and B are each transformed into a sequence of time intervals where the values are above a given threshold τ.

This new query type is very useful for several pharmacogenomics applications. The most straightforward application of threshold queries is the search for similar time series. For example, a common task in pharmacogenomics is the identification of individual drug response or the analysis of the impact of certain environmental influences on gene expression levels or blood values. For this task, the concentration of agents that are suspected to trigger the relevant biochemical reactions is measured over some period of time. Using threshold queries, one is able to efficiently retrieve time series, that are similar to the stimulus time series in terms of threshold crossing events. Note that our technique is able to cope with different thresholds for the stimulus time series and the reacting time series. This is important since usually values of different domains (e.g. chemical concentrations versus gene expression levels) are compared. Another important example where the identification of similar time series is crucial, is the search for similar gene expression patterns. In a time series of gene expression values one can retrieve genes with similar expression levels in order to find genes that are coregulated or showing an interesting response to an external stimulus. In addition, threshold queries can be performed on mixed-type data. Thus, we can correlate data on specific agents such as blood parameter concentrations with gene expression data. Taking an agent time sequence as query object, we can identify genes that are affected by this agent.

In this paper, we propose techniques in order to support these important applications. In particular, we introduce the novel concept of threshold similarity and threshold queries. We propose a suitable data representation method to efficiently support threshold queries. In addition, we present a semi-supervised version of threshold queries which beneficially supports the parameter setting by applying training datasets. Semi-supervised threshold queries automatically detect the best parameter setting for the query process based on a labeled training dataset.

2. Related Work

In general, a time series of length d can be viewed as feature vector in a d-dimensional space, where the similarity between two time series corresponds to their distance in the feature space. Since d is usually large, the analysis of time series data based on

the entire time series information is usually very limited. Due to the so-called *curse of dimensionality*, the efficiency and the effectiveness of data analysis methods decrease rapidly with increasing data dimensionality. Thus, it is mandatory to find more suitable representations of time series data for analysis purposes, e.g. by reducing the dimensionality. In the different communities, several solutions have been proposed. Most of them are based on the following indexing approach: extract a few key *features* for each time series and map each time sequence X to a point $f(X)$ in a lower dimensional feature space, such that the (dis)similarity between X and any other time series Y is approximately equal to the Euclidean distance between the two points $f(X)$ and $f(Y)$. For an efficient access any well known spatial access method can be used to index the feature space.

The proposed methods mainly differ in the representation of the time series: for details see the database-oriented survey[11] and the bioinformatics-oriented survey.[3] The database and the bioinformatics communities have successfully applied standard techniques for dimension reduction to similarity search and data mining in time series databases, including Discrete Fourier Transformation[1] and extensions,[13] Discrete Wavelet Transformation,[6] Piecewise Aggregate Approximation,[14] Singular Value Decomposition,[12,2] Adaptive Piecewise Constant Approximation,[11] Chebyshev Polynomials,[5] cubic splines.[4]

However, all techniques which are based on dimension reduction cannot be applied to threshold similarity queries because necessary temporal information is lost. Usually, in a reduced feature space, the original intervals indicating that the time series is above a given threshold cannot be generated. In addition, the approximation generated by dimensionality reduction techniques cannot be used for our purposes directly because they still represent the exact course of the time series rather than intervals of values above a threshold.

The most important issue for any data analysis purposes is the definition of similarity. The most common way to model (dis-)similarity of feature vectors is to measure their Euclidean distance. For many applications, the Euclidean distance may be too sensitive to minor distortions in the time axis. It has been shown that Dynamic Time Warping (DTW), which is conceptually similar to sequence alignment, can fix this problem.[11] Other common distance functions are Pearson's correlation coefficient which measures the global correlation between two time series, or angular separation, also known as cosine distance, which defines the distance in terms of the angle between two feature vectors. All these distance measures are not directly applicable to threshold similarity queries because all of them consider the absolute values of the time series rather than the intervals of values above a given threshold.

3. Threshold Queries

In this section, we introduce the novel concept of threshold queries based on a similarity model which is very promising for the analysis of pharmacogenomics time series data. Furthermore, we present techniques allowing an efficient query processing.

We define a time series X as a sequence of pairs $(x_i, t_i) \in \mathbb{R} \times T : (i = 1..N)$, where T denotes the domain of time and x_i denotes the measurement corresponding to time t_i. Furthermore, we assume that the time series entities are given in such a way that $\forall i \in 1, .., N-1 : t_i < t_{i+1}$. In most cases, when measuring continuously varying

attributes at discrete time points, the missing values between two observations are estimated by means of interpolation.[3] In the rest of this paper, if not stated otherwise, $x(t) \in \mathbb{R}$ denotes the (interpolated) time series value of time series X at time $t \in T$.

3.1. *Threshold-Crossing Time-Intervals*

Instead of using time series for the description of the time dependent behavior of pharmacogenomics data, we use sequences of time intervals which are related to a specific user-defined threshold, i.e. for a given threshold τ, the pharmacogenomics data is described by means of disjoint time intervals expressing the points of time when the data values are above τ. We call this description of the time dependent behavior *threshold-crossing time-intervals*.

Definition 3.1. Let $X = \langle (x_i, t_i) \in \mathbb{R} \times T : i = 1..N \rangle$ be a time series with N measurements and $\tau \in \mathbb{R}$ be a threshold. Then the *threshold-crossing time interval sequence* of X with respect to τ is a 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.

The description of the time dependent behavior of some attribute by means of threshold-crossing time-intervals is a simple method, but generally very promising for the analysis of pharmacological data, in particular for the analysis of pharmacogenomics data. The advantages compared to common methods which consider the entire time series are as follows: time-interval sequences are easy to handle and can be processed and stored very efficiently. Furthermore, threshold-crossing time-interval sequences express the temporal behavior w.r.t. a certain threshold, i.e. the intervals denote the time points when the attribute is above or below a specific threshold value.

The latter advantage is most decisive because threshold-crossing time-interval sequences allow us to infer a causality from the temporal behavior for relevant threshold levels instead of considering the entire time series curve. Let us assume that we have to analyze the temporal behavior of gene expression values by means of fluorescence intensity measurements. Measurable fluorescence intensities are supposed to be proper expression values plus noises. One can imagine that the measurements in the medial florescence spectrum are more precisely compared to the measurements in the extreme spectral areas, because the sensor data from the extreme spectral areas have more noise than the data from the medial spectrum. Traditional analyzing approaches which consider the entire expression level range can lead to incorrect results due to noise.

3.2. *Similarity Model*

Intuitively, two time intervals are defined to be similar if they have "similar" starting and end points, i.e. they are starting at similar times and ending at similar times.

Definition 3.2. Let $t1 = (t1_l, t1_u) \in T \times T$ and $t2 = (t2_l, t2_u) \in T \times T$ be two time intervals. Then the distance function $d_{int} : (T \times T) \times (T \times T) \rightarrow \mathrm{R}$ between two time intervals is defined as:

$$d_{int}(t1, t2) = \sqrt{(t1_l - t2_l)^2 + (t1_u - t2_u)^2}$$

Since for a certain threshold τ a time series object is represented by a sequence or a set of time intervals, we need a distance/similarity measure for sets of intervals. Several distance measures for set based objects have been introduced in the literature.[7] The Sum of Minimum Distances (SMD) most adequately reflects the intuitive notion of similarity between two threshold-crossing time interval sequences. According to the SMD we define the *threshold-distance* d_{TS} as follows:

Definition 3.3. Let X and Y be two time series and $S_X = TCT_\tau(X)$ and $S_Y = TCT_\tau(Y)$ be the corresponding threshold-crossing time interval sequences.

$$d_{TS}(S_X, S_Y) = \frac{1}{|S_X|} \cdot \sum_{s \in S_X} \min_{t \in S_Y} d_{int}(s,t) + \frac{1}{|S_Y|} \cdot \sum_{t \in S_Y} \min_{s \in S_X} d_{int}(t,s),$$

The idea of this distance function is to map every interval from one sequence to the closest (most similar) interval of the other sequence and vice versa. Let us note that the threshold-distance between two time series according to a certain threshold τ is also called τ-similarity.

We are now able to define the *threshold query* which reports the k most τ-similar time series of a database DB for a given query time series Q.

Definition 3.4. Let DB be the domain of time series objects. The threshold query consists of a query time series $Q \in DB$ and a query threshold $\tau \in \mathbb{R}$. The threshold query reports the smallest set $TSQ_k(Q,\tau) \subseteq DB$ of time series objects that contains at least k objects from DB such that

$$\forall X \in TSQ_k(Q,\tau), \forall Y \in DB - TSQ_k(Q,\tau) :$$

$$d_{TS}(TCT_\tau(X), TCT_\tau(Q)) < d_{TS}(TCT_\tau(Y), TCT_\tau(Q)).$$

3.3. *Threshold Invariant Data Representation*

In this section, we introduce a novel data representation of the time series objects. Although the time series objects may be very complex, i.e. contain lots of measured values, we need a data representation of the time series objects which allows us to process the threshold queries very efficiently.

The basic idea of our proposed time series representation is that we do not need to access the complete time-series data at query time. Instead, only partial information of the time-series objects which can be efficiently accessed might suffice to report the results. In order to achieve these requirements, the threshold-crossing time-interval-sequences of all possible thresholds must be pre-computed and materialized which is fulfilled by our novel data representation. The key for the novel time series representation is a decomposition of a time series into a set of trapezoids, as depicted in Figure 2. The upper and lower edge of each trapezoid is parallel to the time axis, the left side is bounded by an increasing time series segment and the right side is bounded by a decreasing segment. Each trapezoid represents a set of threshold-crossing time intervals which correspond to a certain range of threshold values. The complete set of required threshold-crossing time intervals w.r.t. an arbitrary threshold τ can be easily computed by all trapezoids crossing τ. We have developed an algorithm which decomposes time series into corresponding trapezoids in linear time w.r.t. the length of the time series, due to the natural ordering of the time series values.

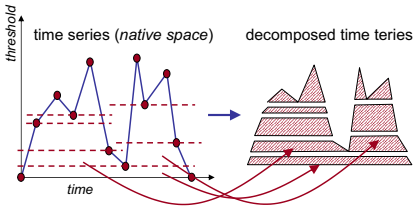


Figure 2. Time Series Decomposition

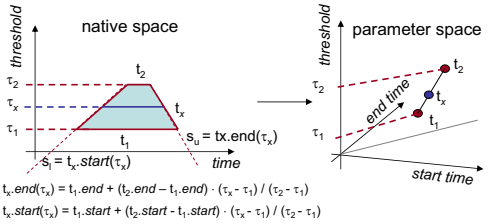


Figure 3. Interval Ranges in Parameter Space

For the management of the trapezoids we transform the trapezoids into a three dimensional space ($start \in T, end \in T, \tau \in \mathbb{R}$), where *start* denotes the start time and *end* denotes the end time for all threshold-crossing time-intervals and τ denotes the corresponding threshold. In the following, we will call this space *parameter space*. A 2-dimensional plane along the threshold axis parallel to the (start,end)-plane at a certain threshold τ in the parameter space is called *time-interval plane* of threshold τ. The time-interval plane has some advantages for the efficient management of intervals. First, the time distances between intervals are preserved. Second, the position of large intervals, which are located within the upper-left region, substantially differs from the position of small intervals. However, the most important advantage is that in this space the Euclidean distance correspond to the (dis-)similarity of intervals according to Definition 3.2.

For the efficient management of the trapezoids we use the following observation: all threshold-crossing time-intervals $tct_{\tau_i,j}(X)$ which start at segment s_l and end on segment s_u lie in the parameter space on the three-dimensional straight line: $g_P : \overrightarrow{x} = \overrightarrow{p_1} + \triangle t \cdot (\overrightarrow{p_2} - \overrightarrow{p_1})$, where $\overrightarrow{p_1} = (tct_{\tau_1,1}(X).start, tct_{\tau_1,1}(X).end, \tau_1)^T$ and $\overrightarrow{p_2} = (tct_{\tau_2,2}(X).start, tct_{\tau_2,2}(X).end, \tau_2)^T$. An example is depicted in Figure 3. At query time, the time-interval plane coordinates of the threshold-crossing time-intervals which correspond to the query threshold τ_q can be easily determined by computing the intersection of all segments of the parameter space with the time-interval plane P at threshold τ_q.

3.4. *Indexing Segments of the Parameter Space*

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 3-dimensional segments by rectangles. The R*-tree efficiently support nearest-neighbor queries[9] which build the basis of the computation of the similarity between two time series objects.

4. Semi-Supervised Threshold Queries

In the previous section (Section 3.3) we proposed a threshold invariant data representation which allows to perform efficient threshold queries where the threshold can be chosen at query time. The question is now, which threshold should be chosen for any analyzing task?

The answer to that question not only depends on the specific characteristics of the data at hand. Quite often the user expects a certain query to yield certain results. Then the first interesting task is to determine a threshold value that yields good results.

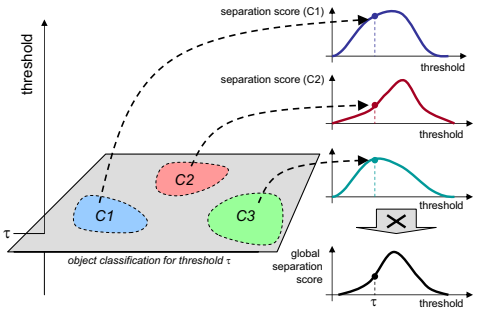


Figure 4. Determination of the threshold dependent class separation score.

Furthermore our experimental studies revealed that a shift in the user's expectation frequently makes it necessary to readjust the threshold value the similarity measure is based on.

In the following, we assume that the user's expectation is modeled by means of a small training data set. Results for a given query are considered good, if a lot of results are marked with the same class label as the query time series. The general idea of our technique is to search for threshold values that promise high similarity scores between objects with similar behavior and low scores between objects with different behavior. In the next sections we will explain how we derive quality values for different thresholds and how we use this information to obtain global high quality values.

4.1. *Threshold Value Quality for One Class*

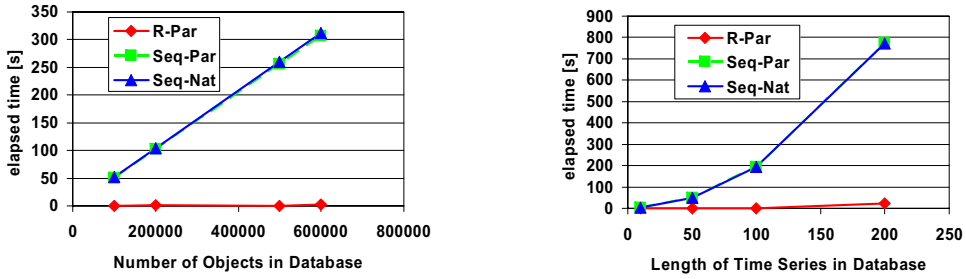
In this section we will outline how we detect suitable threshold values for a fixed class. Let us assume we are given a training data set C_M which consists of k classes, $C_M = C_1, ..., C_k$. For a fixed class C_i, we need to determine these threshold values which yield a good separability of C_i from the remaining classes. To evaluate the score of a query for a fixed class C_i and a fixed threshold τ, we pairwise compute the silhouette width[10] of C_i and all other classes C_j, $j \neq i$, in C_M. The minimal silhouette width of all pairs is used as separation score of a threshold τ. Calculating the separation score for any existing threshold results in a quality measure which estimates the separability score for a training data set C_M, a class $C_i \in C_M$ and a given threshold τ.

4.2. *Derivation of a Global Suitable Threshold Value*

In the last section, we have developed a quality measure which computes the separation score for each class C_i of our training data set C_M. Now, we need a suitable combination of all k separation score functions. For our approach, we chose the sum of all score functions, i.e. compute the silhouette coefficient[10] for C_M. The global separation score function now reflects the overall separability score of our training data set for an arbitrary threshold τ. Based on the idea of semi-supervised learning, the global score function gives the user hints to chose the most promising threshold ranges. The example depicted in Figure 4 points up one step of this procedure for a certain threshold.

5. Evaluation

In this section we will present the results of our experiments with respect to efficiency and effectiveness.



(a) Scalability against database size.

(b) Scalability against time series length.

Figure 5. Efficiency Evaluation.

5.1. *Efficiency*

In this section, we present the results of a large number of experiments performed on a selection of different time series datasets. In particular, we compared the efficiency of our proposed approach (in the following denoted by 'R_{Par}') for answering threshold queries using one of the following techniques. The first competing approach works on native time series. At query time for each database time series the corresponding threshold-crossing time intervals (TCT) are computed for the query threshold and afterwards the distance between the query time series and the corresponding database object are derived. In the following this method will be denoted by 'Seq_{Nat}' as it corresponds to a sequential processing of the entire set of the native data. The second competitor works on the parameter space rather than on the native data. It stores all TCTs without using any index structures. As this storage leads to a sequential scan over the corresponding elements of the parameter space at query time, we will refer to this technique as the 'Seq_{Par}' method.

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. 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. Figure 5(a) exhibits the results for each database size averaged over several thresholds and several randomly chosen queries. Second, we explored the impact of the length of the time series to be compared. The results depicted in Figure 5(b) show that our approach significantly outperforms its competitors and is scalable w.r.t. the number and the length of the time series objects.

5.2. *Effectiveness*

At first, we exemplarily show how threshold queries can be beneficially used for pharmacogenomics in practice. An important application of threshold queries is finding genes of similar function or of similar drug response in gene expression data. We used the yeast gene expression time course data sets from Gene Expression Omnibus[a] with accession numbers "GDS30" and "GDS38" and launched for each gene of each data

[a]http://www.ncbi.nlm.nih.gov/geo/

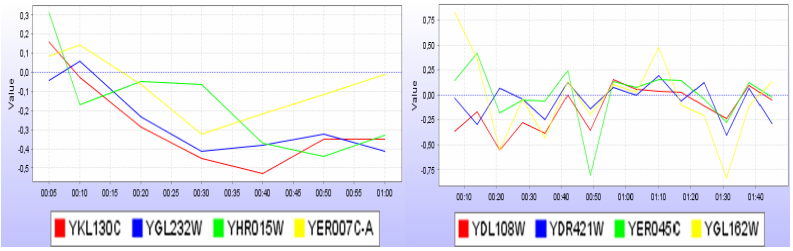


Figure 6. Illustration of sample threshold query results (best seen in color).

set a threshold query against the rest of the genes. For each gene, we thus computed its three most similar genes based on threshold similarity on each data set. We evaluated the gene functions using Gene Ontology (GO).[b] The results of two sample queries are depicted in Figure 6 (absolute query threshold $\tau = 0$). On the left side of the figure, the 3 most similar genes to ORF "YKL130C" are depicted. All three reported ORFs "YGL232W", "YHR015W", and "YER007C-A" code for proteins with RNA-binding activity. On the right hand side of Figure 6 the 3 most similar genes to ORF "YDL108W" are depicted. All three reported ORFs "YDR421W", "YER045C", and "YGL162W" code for proteins with RNA polymerase II transcription factor activity.

In the example above no information about the dataset characteristics was used to select the threshold value. Different settings of the threshold value can lead to different qualities of the results. In the next experiment, we investigate the effectiveness of semi-supervised threshold queries which are used to find the optimal threshold value by means of a training dataset. At first, we are interested in how the optimal threshold values change when the expected results change, i.e. when the focus of the query changes. The following experiments were performed on the data sets "GDS30" and "GDS38" from Gene Expression Omnibus. For the first experiment we used the GO functional classes on level 4. Afterwards we changed the focus of our queries to the GO level 5. As expected, we obtained different optimal threshold values (cf. Figure 7).

These results gave raise to the question whether the computed optimal threshold values do indeed yield good results on the whole data set. To evaluate this, we clustered the time series for varying threshold values and determined the rand index.[8] For example, the threshold value 0.73 which corresponds to a high separation score on the GDS30 data set for GO level 4 resulted in a rand index equal to 0.94. Contrary, when using a threshold value of 0.2 the rand index decreased to 0.86. Similar results were observed for other levels, for other threshold values, and on other datasets.

6. Conclusions

In this paper, we proposed the novel concept of semi-supervised threshold queries which are suitable to analyze time series data in the area of life science. In addition, we presented a data representation method that supports threshold queries efficiently. In our experimental evaluation, we have shown that our proposed data representation accelerates threshold queries drastically. Furthermore, the results of our experimental evaluations have shown that the proposed semi-supervised threshold queries are very

[b] http://www.geneontology.org/

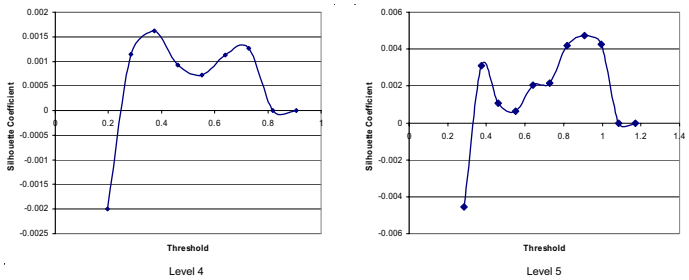


Figure 7. Different Optimal Threshold Values for Different GO Levels.

effective and are very useful in particular for the analysis of pharmacogenomics time series databases. For future work, we plan to extend our approaches to data mining tasks, such as similarity join and clustering.

Acknowledgments

Part of this work is supported by the German Ministry for Education, Science, Research and Technology (BMBF) under grant no. 031U112F within the BFAM (Bioinformatics for the Functional Analysis of Mammalian Genomes) project.

References

1. R. Agrawal, C. Faloutsos, and A. Swami. "Efficient Similarity Search in Sequence Databases". In *Proc. 4th Conf. on Foundations of Data Organization and Algorithms*, 1993.
2. O. Alter, P. Brown, and D. Botstein. "Generalized Singular Value Decomposition for Comparative Analysis of Genome-Scale Expression Data Sets of two Different Organisms". *Proc. Natl. Aca. Sci. USA*, 100:3351–3356, 2003.
3. Z. Bar-Joseph. "Analyzing Time Series gene Expression Data". *Bioinformatics*, 20(16):2493–2503, 2004.
4. Z. Bar-Joseph, G. Gerber, T. Jaakkola, D. Gifford, and I. Simon. "Continuous Representations of Time Series Gene Expression Data". *J. Comput. Biol.*, 3-4:341–356, 2003.
5. Y. Cai and R. Ng. "Index Spatio-Temporal Trajectories with Chebyshev Polynomials". In *Proc. ACM SIGMOD*, 2004.
6. K. Chan and W. Fu. "Efficient Time Series Matching by Wavelets". In *Proc. IEEE ICDE*, 1999.
7. T. Eiter and H. Mannila. "Distance Measure for Point Sets and Their Computation". In *Acta Informatica, 34*, pages 103–133, 1997.
8. M. Halkidi, Y. Batistakis, and M. Vazirgiannis. "On Clustering Validation Techniques". In *Intelligent Information Systems Journal*, 2001.
9. G. Hjaltason and H. Samet. "Ranking in Spatial Databases". In *Proc. Int. Symp. on Large Spatial Databases (SSD'95), Portland, OR*, 1995.
10. L. Kaufman and P. Rousseeuw. "Finding Groups in Data: An Introduction to Cluster Analysis". Wiley, New York, 1990.
11. E. Keogh, K. Chakrabati, S. Mehrotra, and M. Pazzani. "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases". In *Proc. ACM SIGMOD*, 2001.
12. F. Korn, H. Jagadish, and C. Faloutsos. "Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences". In *Proc. ACM SIGMOD*, 1997.
13. S. Wichert, K. Fokianos, and K. Strimmer. "Identifying Periodically Expressed Transcripts in Microarray Time Series Data". *Bioinformatics*, 20(1):5–20, 2004.
14. B. K. Yi and C. Faloutsos. "Fast Time Sequence Indexing for Arbitrary Lp Norms". In *Proc. VLDB*, 2000.