Multi-resolution Approach to Time Series Retrieval

Muhammad Marwan Muhammad Fuad, Pierre-François Marteau

Multi-resolution Approach to Time Series Retrieval

Muhammad Marwan Muhammad Fuad
VALORIA, Université de Bretagne Sud
Université Européenne de Bretagne
BP 573, 56017 Vannes
marwan.fuad@univ-ubs.fr

Pierre-François Marteau
VALORIA, Université de Bretagne Sud
Université Européenne de Bretagne
BP 573, 56017 Vannes
pierre-francois.marteau@univ-ubs.fr

ABSTRACT
We propose a new multi-resolution indexing and retrieval method of the similarity search problem in time series databases. The proposed method is based on a fast-and-dirty filtering scheme that iteratively reduces the search space using several resolution levels. For each resolution level the time series are approximated by an appropriate function. The distance between the time series and the approximating function is computed and stored at indexing-time. At query-time, assigned filters use these pre-computed distances to exclude wide regions of the search space, which do not contain answers to the query, using the least number of query-time distance computations. The resolution level is progressively increased to converge towards higher resolution levels where the exclusion power rises, but the cost of query-time distance computations also increases. The proposed method uses lower bounding distances, so there are no false dismissals, and the search process returns all the possible answers to the query. A post-processing scanning on the candidate response set is performed to filter out any false alarms and return the final response set. We present experimentations that compare our method with sequential scanning on different datasets, using different threshold values and different approximating functions. The experiments show that our new method is faster than sequential scanning by an order of magnitude.

Categories and Subject Descriptors
H.3.3 [**Information Systems**]: Information Search and Retrieval – Search process, Retrieval models.

Keywords
Time Series Information Retrieval, Multi-resolution, Sequential Scanning, Metric Spaces, MIR.

1. INTRODUCTION
Time series similarity search is a fundamental problem in computer science. This problem has many medical, financial, and scientific applications. Similarity between two time series can be depicted using a *similarity distance*. Time series similarity search can be viewed as retrieving all the time series in the database that are "near" a given query according to a given similarity distance.

Time series data have high complexity. This high complexity can be reduced by time series representation methods which aim at reducing the high complexity by transforming the time series into a lower dimensional space and performing the similarity search process at these lower dimensional spaces.

There have been different suggestions to represent time series in lower dimensional spaces, to mention a few: *Discrete Fourier Transform* (DFT) [1,2] , *Discrete Wavelet Transform* (DWT) [4], *Singular Value Decomposition* (SVD) [7], *Adaptive Piecewise Constant Approximation* (APCA) [6], *Piecewise Aggregate Approximation* (PAA) [5,11], *Piecewise Linear Approximation* (PLA) [9], *Chebyshev Polynomials* (CP) [3].

Time series representation methods are based on predefined schemes, in that the parameters which control their performance have been decided at indexing-time, and the search process depends highly on these parameters which may prove to be inappropriate at query-time.

In this work we present a new scheme that offers more control on these indexing-time parameters. This control enables the search algorithm to make the necessary computations only, starting with less costly computations, whose exclusion power is lower, and moving to more expensive computations, with more exclusion power, only when the less costly computations fail to exclude the time series.

This paper is organized as follows: the related background is presented in section 2. In section 3 we present the new algorithm. The experiments we conducted are presented in section 4. In section 5 we discuss some of the results of the experiments, and we conclude the paper with section 6

2. RELATED WORK
2.1 Metric Spaces
Let D be a set of objects. A function $d : D \times D \rightarrow \Re^{+} \cup \{0\}$ is called a distance metric if the following holds;

i- $d(x, y) \geq 0$ (non-negativity)

ii- $d(x, y) = d(y, x)$ (symmetry)

IDEAS10 2010, August 16-18, Montreal, QC [Canada].
Editor: Bipin C. DESAI

iii- $x = y \Leftrightarrow d(x,y) = 0$ (identity)

iv- $d(x,z) \le d(x,y) + d(y,z)$ (triangular inequality)

$\forall x, y, z \in D$. We call (D, d) a metric space

2.2 Range Queries

Given a query Q and a radius r, which represents a *threshold,* a *tolerance*, or a *selectivity*. The range query problem can be specified as retrieving all the data objects that are within a distance r from that query. This can be represented as:

$$RQ(Q,r) = \{S \in O \ ; d(Q,S) \le r\} \quad (1)$$

where O is the set of objects in the database

2.3 Sequential Scanning

In many applications distance computations can be a time consuming task that other tasks such as CPU time or even I/O time can be neglected. For this reason search algorithms try to avoid distance computations as much as possible. In fact, in many cases the performance of an algorithm is measured by the number of distance computations it requires. Different distances also take different computing times. For instance, the *dynamic time warping* (DTW), which is a similarity measure that is widely used in time series databases, gives better results in time series data mining tasks than the Euclidean distance, but it is more costly to compute.

A trivial solution to the similarity search problem is sequential scanning, also known as *linear scanning*, where the query is compared against all the data objects in the data base. So in order to perform a range query (Q,r), the distance between the query Q and all the data objects in the data base is computed, and all the data objects that satisfy $d(Q,S) \le r$ constitute the response set.

It is easy to notice that in the case where the size of the data base is very large, which is the case with most databases in use today, sequential scanning is not the best scenario to answer similarity queries because it requires too many distance evaluations.

Different techniques can be used to avoid the high cost of sequential scanning. One of them is using indexing structures, which are offline procedures based on storing some distance calculations. Later, at query-time, these calculations can be used to exclude the time series which, according to these pre-computed distances, can not be answers to the query. This is what we call a *fast-and-dirty* filtering of data. What remains of the time series is scanned sequentially against the query to get the final answers to the query. A fast-and-dirty filter was presented in [8]

However, even these structures can fail in handling high-dimensional databases that their performance can deteriorate to become similar to that of sequential scanning, or even worse. This is what we call the *dimensionality curse*.

2.4 Representation Methods

Managing high-dimensional time series is not a trivial problem. Time series are highly correlated data, so representation methods use a scheme that aims at reducing the dimensionality of the time series by projecting the original data onto lower dimensional spaces and processing the query in those reduced spaces. This scheme is widely used in time series data mining community.

When embedding the original space into a lower dimensional space and performing the similarity query in the transformed space, two main side-effects may be encountered; *false alarms*, and *false dismissals*. False alarms are time series that belong to the response set in the transformed space, but do not belong to the response set in the original space. False dismissals are time series that the search algorithm excluded in the transformed space, although they are answers to the query in the original space. Generally, false alarms are more tolerated than false dismissals, because a post-processing scan is usually performed on the results of the query in the transformed space to filter out these time series that are not valid answers to the query in the original space. However, false alarms can slow down the search time if the algorithm returns too many of them. False alarms and false dismissals are dependent on the transformation used in the embedding.

If f is a transformation from the original space (O_{orig}, d_{orig}) into another space (O_{trans}, d_{trans}) then in order to guarantee no false dismissals this transformation should satisfy:

$$d_{trans}(f(u_1), f(u_2)) \le d_{orig}(u_1, u_2), \quad \forall u_1, u_2 \in O_{orig} \quad (2)$$

The above condition is known as the *lower-bounding lemma* .

3. THE PROPOSED METHOD

3.1 Motivation

Time series representation methods have the following scheme: choose a lower dimensional space, represent the time series in the reduced space, define a lower bounding similarity distance on this reduced space , process the similarity search in the reduced space , exclude the time series which are farther than r from the query. At the end, we get a *candidate answer set*, scan this candidate answer set using the original time series and the original similarity distance to obtain the final answer set.

The problem with this approach is that it uses a one-phase scheme. The dimension of the reduced space is decided at indexing-time and the performance at query-time depends completely on the choice made at indexing-time. But in practice, we do not know a priori the optimal dimension of the reduced space.

In this work, we try to address this problem differently by establishing a model that involves a multi-resolution representation of time series; we use several reduced spaces, or as we call them *resolution levels*. The indexing system stores different numbers of pre-computed distances, related to the number of resolution levels. Lower resolution levels have lower dimensions, so distance computations at these levels are less

costly than higher resolution levels where dimensions are higher, so distance evaluations are more expensive. But the computation complexity at any level is always less expensive than the computation complexity of sequential scanning, because even at the highest level, the dimension is still lower than that of the original space, which is used in sequential scanning. In our method, the search algorithm starts with the lowest resolution level, and tries to exclude the time series, which are not answers to the query, at that level where the distances are not costly to calculate, and the algorithm does not access a higher level until all the pre-computed distances of the lower level have been exploited. We call our method the *Multi-resolution Indexing and Retrieval* scheme (MIR).

3.2 Concepts and Terminology

Let O be the original n-dimensional space where the time series are embedded, R is a $2m$-dimensional space, where $2m \le n$. Each time series $S \in O$ is divided into m segments, each of which is approximated by a function of low dimension: a polynomial of degree (1:5), for instance, where the degree of this approximating function is lower than the length of the segments, and where the approximation error, according to a given distance, between this segment and the approximating function is minimal, so this function is the optimal approximation of that segment. A polynomial of the same degree is used to approximate all the segments of all the time series in the database.

We associate every segment with two related concepts; the first is the image of all the points of that segment on the approximating function. The *image vector* $\tilde{S}$ is, by definition, an n-dimensional vector whose components are the images of all the points of all the segments of that times series. The second concept is the images of the two end points of that segment on the approximating function, which we call the *main image* of that segment. So for a time series of m segments we have $2m$ main images. Those $2m$ main images are, by definition, the *projection vector* S^R of the time series on R. Figure 1 illustrates the different definitions we presented in this section: The segment [0:3] is approximated by a first-degree polynomial. The image of this segment is the points [*a*, *b*, *c*, *d*]. The main image of this segment is the points [*a*, *d*].

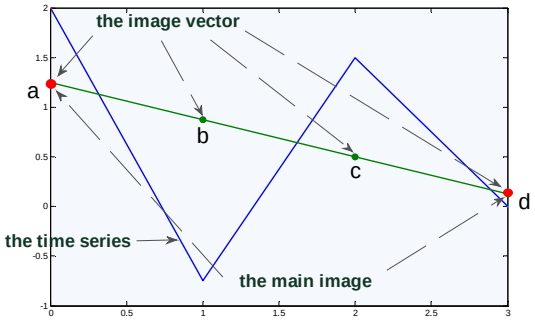

Figure 1: The different concepts of the proposed method

We define two distances on the database, the first is denoted by d, and is defined on an n-dimensional space, so it is the distance between two time series in the original space , i.e. $d(S_i, S_j)$, or the distance between the original time series and its image vector, i.e. $d(S_i, \tilde{S}_i)$. We choose d to be Euclidean (or Minkowski distance, in general), thus d is metric.The second distance is denoted by d^R, and is defined on a $2m$-dimensional space, so it is the distance between two projection vectors, i.e. $d^R(S_i^R, S_j^R)$.

Notice that since the main image of each segment is a partial set of the image of that segment, this implies that the components of the projection vector form a partial set of the components of the image vector. Consequently, the distance d^R is a partial distance of d. The direct result of this is that when we use the Euclidean distance (or any Minkowski distance), for both d and d^R we get:

$$d^R(S_1^R, S_2^R) \le d(\tilde{S}_1, \tilde{S}_2) \qquad (3)$$

Relation (3) means that d^R is lower bounding of d .

The *resolution level* k is an integer related to the dimensionality of the reduced space R. So the above definitions of the projection vector and the image vector can be extended to a further segmentation of the time series, with different values $m \le m_k$, The image vector and the projection vector at level k are denoted by $\tilde{S}^{(k)}$ and $S^{R(k)}$, respectively. Figure 2 shows an example of the relationships between the previous concepts.

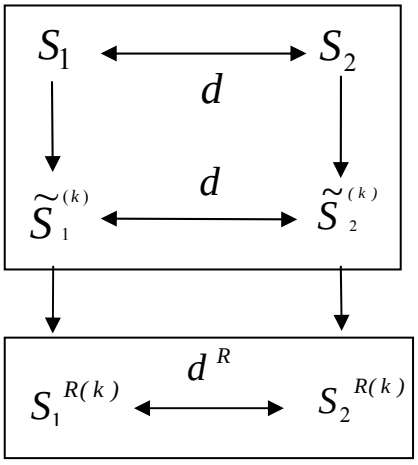

Figure 2: The original space *(O, d)* embeds the original time series S_1, S_2 (top), the images of the approximated time series $\tilde{S}_1, \tilde{S}_2$ (middle). The reduced space *R* embeds the main images of the approximated time series S_1^R, S_2^R (bottom).

3.3 The Algorithm Description

Given a query (Q, r), let $\tilde{S}^{(k)}, \tilde{Q}^{(k)}$ be the projection vectors of S , Q , respectively, on their approximating functions, where

S is a time series in the database. By applying the triangular inequality we get:

$$d(\widetilde{Q}^{(k)}, S) \le d(Q,S) + d(Q,\widetilde{Q}^{(k)}) \quad \forall S \in O \qquad (4)$$

Since we assumed that $\widetilde{S}^{(k)}$ is the best approximation of S at level k, then for any $S \in O$ we have: $d(\widetilde{Q}^{(k)}, S) \ge d(S, \widetilde{S}^{(k)})$.

This means that all the time series that satisfy:

$$d(S, \widetilde{S}^{(k)}) > r + d(Q, \widetilde{Q}^{(k)}) \qquad (5)$$

Should be excluded.

In a similar way, by applying the triangular inequality, and taking into consideration that $\widetilde{Q}^{(k)}$ is the best approximation of Q at level k, we can safely exclude all the time series that satisfy:

$$d(Q, \widetilde{Q}^{(k)}) > r + d(S, \widetilde{S}^{(k)}) \qquad (6)$$

(5) and (6) can be written in one relation:

$$\left| d(Q, \widetilde{Q}^{(k)}) - d(S, \widetilde{S}^{(k)}) \right| > r \qquad (7)$$

Inequality (7) defines the first exclusion condition, which we call *the first filter*

On the other hand, we have:

$$d(\widetilde{S}^{(k)}, \widetilde{Q}^{(k)}) \le d(\widetilde{Q}^{(k)}, S) + d(S, \widetilde{S}^{(k)}) \qquad (8)$$

Using the triangular inequality again and taking (3) into consideration we get:

$$d^R(S^{R(k)}, Q^{R(k)}) > r + d(Q, \widetilde{Q}^{(k)}) + d(S, \widetilde{S}^{(k)}) \qquad (9)$$

We call the above exclusion condition *the second filter* □

The Indexing-time: The application of our method starts by choosing the length of segments for each resolution level. There is no optimal choice of lengths, so we choose lengths which are of power of 2. The shortest length corresponds to the lowest level, and the longest corresponds to the highest level. Then we choose the approximating function to be used with all the time series and for all resolution levels. This means that if we choose to use a polynomial of the first degree, then all the segments of all the times series in the database, and for all resolution levels, should be approximated using a polynomial of the first degree. Our method works with any polynomial, or even any other approximating function. However, in this paper we use polynomials for their simplicity.

We compute and store all the distances $d(S, \widetilde{S}^{(k)}) \quad \forall S \in O$

The Query-time: The query is divided into segments with the same lengths as those of the time series and for each resolution level. These segments are approximated using an approximating function of the same type that was used to approximate the time series. The distances $d(Q, \widetilde{Q}^{(k)})$ are computed. Notice that $d(Q, \widetilde{Q}^{(k)})$ are computed only once for all the time series in the database.

At each resolution level, the first filter is less costly to apply than the second filter, because it does not include any distance evaluation, since the two distances it uses have already been computed at indexing-time. The second filter contains two distances that have been computed at indexing-time ($d(Q, \widetilde{Q}^{(k)}), d(S, \widetilde{S}^{(k)})$), so the only distance that is to be computed at query-time is $d^R(S^{R(k)}, Q^{R(k)})$. Since lower resolution levels have lower dimensions, the second filter is less costly to compute at those levels than at higher levels, where the dimensionality increases. But at any level, the cost of applying the second filter is never as costly as the distance computations at the original space, because we assumed that $2m \le n$.

We start with the lowest level and try to exclude the first time series using (7). If this time series is excluded, we move to the next time series, if not, we try to exclude this time series using relation (9). If all the time series in the database have been excluded the algorithm terminates, if not, the algorithm moves to a higher level. Finally, after all levels have been exploited, we get a candidate answer set, which is sequentially scanned to filter out any false alarms and obtain the final answer set.

So the proposed algorithm does not compute a more expensive distance calculation unless it has tried to exclude the time series using a less expensive distance at a lower resolution level.

4. EXPERIMENTS

We conducted extensive experiments on different datasets available at UCR [12]. To make sure that our experiments are statistically significant we excluded the datasets that are too small (less than 100 instances). So the datasets we tested have sizes that vary between 100 and 6164 instances. The length of the time series varies between (60) and (463). The approximating functions we used in our experiments were polynomials of the first, third, and fifth degree. The distance we used for both d and d^R was the Euclidian distance. We compared our method with sequential scanning (also with the Euclidean distance) since this is the baseline method. The values of r varied between r that returns 1% of the time series of that dataset (in sequential scanning) and r that returns 10% of the time series. For each dataset and for each value of r we launched the query 100 times and took the average of these 100 runs. The queries in all cases were time series from the dataset chosen at random, then noise was added to them

We opted for a platform-independent approach to test our method using the *latency time* concept obtained from a performance study

of floating point operations [10]. The latency time is based on the number of cycles the processor takes to perform different arithmetic operations, so we added a counter to compute the number of different operations (>,+ - ,*,abs, sqrt) that both sequential scanning and our method used in the search process. Then the number of each operation was multiplied by the latency time of that operation to get the total latency time for sequential scanning and for our method. The latency time is 5 cycles for (>, + -), 1 cycle for (abs), 24 cycles for (*), and 209 cycles for (sqrt). This approach actually puts our method at a disadvantage, because our method uses the square root operation, which is an expensive operation, more often. So the results of the experiments should be viewed as a worst case performance of our method. Figure 3, shows some the results we obtained using as an approximating function a first, third, and fifth degree polynomial, on time series of average length (between 128 and 150)

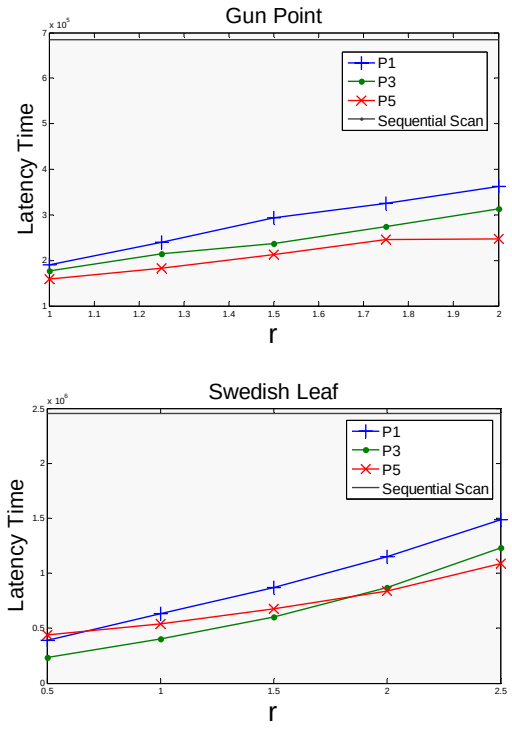

Figure 3: Comparison of the latency time between sequential scanning and MIR on datasets (Gun Point) (above), and (Swedish Leaf) (below). The figure shows the latency time using as an approximating function a polynomial of the first (P1), third (P3), and fifth degree (P5)

The results show that MIR outperforms sequential scanning by an order of magnitude on average.

Comparing the performance with the length of the time series shows that the performance of MIR improves in general as the time series get longer. Figure 4 shows the results we obtained with the two longest time series among the tested datasets. (Yoga) (426) and (Fish) (463).

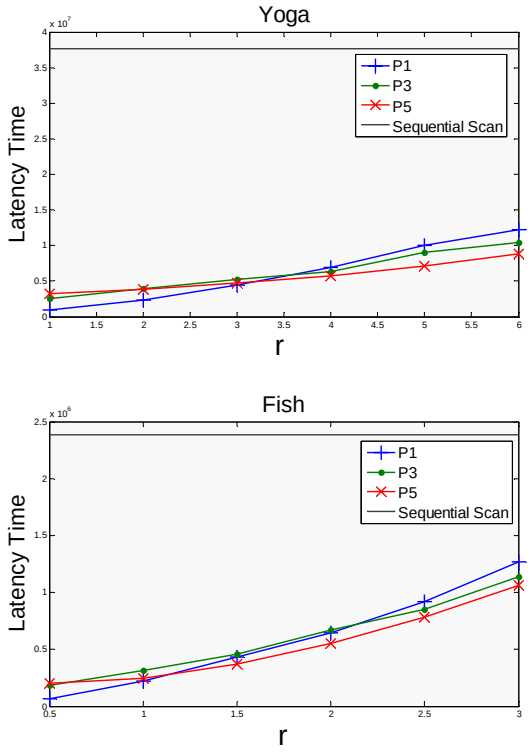

Figure 4: The performance of the two datasets (Yoga) (above) and (Fish) (below); the longest datasets in the UCR archive.

We can also see from Figure 4 that as the times series get longer, the degree of the polynomial has less impact on the performance of the algorithm. In order to examine this phenomenon more closely, we tested MIR on the dataset (motorCurrent) (1500). This dataset is also from the UCR archive (but it is not available online) and it is almost four times as long as the longest online dataset. Figure 5 shows the results we obtained from applying MIR to this dataset.

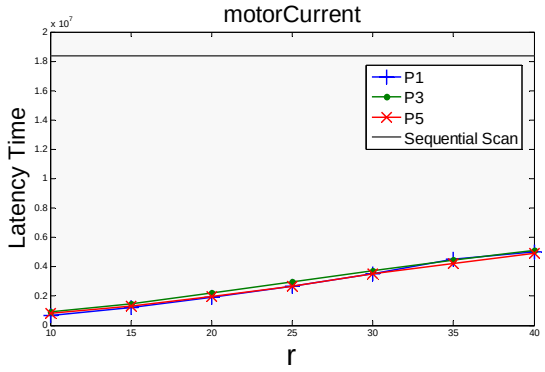

Figure 5: Comparison of the latency time between sequential scanning and MIR on dataset (motorCurrent) (1500). The figure shows the latency time using as an approximating function a polynomial of the first (P1), third (P3), and fifth degree (P5)

These results enhance the two previously stated outcomes; the performance gets better in general as the time series get longer,

and that the influence of the polynomial degree decreases in this case.

We also designed a particular experiment to study the relationship between the length of the time series and the performance of MIR: We chose a particular dataset (also from the UCR but not available online) called (Tickwise). This dataset consists of one extremely long time series (279113). We extracted the first (204800) part of it to construct a dataset of 200 time series each has a length of (1024) (power of 2). We call this dataset (Long Tickwise). We constructed another dataset called (Short Tickwise) which consists of 200 time series each of which is the first (128) part of (Long Tickwise), so the two datasets have the same nature (the same data) and the same size. The only difference is the length of the time series. Figure 6 shows a comparison of the latency time of the two constructed datasets using a first degree polynomial as an approximating function. Since the number of operations of sequential scanning is different Figure 6 shows the proportion of the number of operations that dataset needed to perform the similarity search using MIR to the number of operations the sequential scanning needed to perform the similarity search in that dataset. Notice that the values of r that return 1%-10% of the time series are not the same for the two datasets so the values of r in Figure 6 are superposed. The results clearly show that the performance of MIR improves as the length of time series gets longer for this dataset

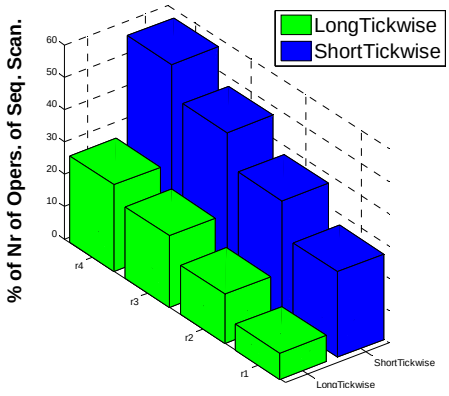

Figure 6: Comparison of the proportion of operations using MIR to the number of operations using sequential scanning in (LongTichwise) with the proportion of operations using MIR to the number of operations using sequential scanning in (ShortTichwise). The approximating function is a first degree polynomial.

The experiments show that the exclusion process depends on the value of r, the resolution level, and the dataset in question. Figure 7 shows the exclusion process for two datasets (Adiac) and (ECG200) for the value of r that returns 1% of the time series. The time series in (Adiac) have a length of (176), so this dataset uses 7 resolution levels, while time series in (ECG200) have a length of (96) so this dataset uses 6 resolution levels. The approximating function is a first degree polynomial.

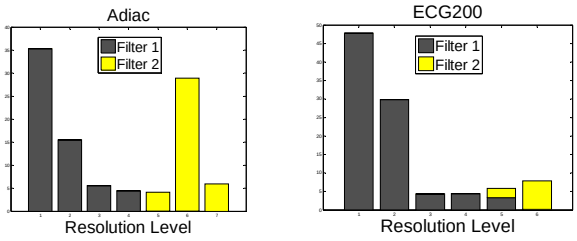

Figure 7: The exclusion process of datasets (Adiac) (left) and (ECG200)(right) for *r* that returns 1% of the time series.

It is important to mention that Figure 7 does not show the exclusion power of each resolution level or each filter but only how these participate in the exclusion process, since at each resolution level the algorithm starts by applying filter one and it applies filter two only on the time series that could not be excluded by filter one. This in fact means that filter two has a higher exclusion power than filter one. Likewise, the algorithm does not move to a resolution level *k* unless it has failed to exclude the time series at resolution level *k-1*, but the exclusion power of level *k* is higher than that of level *k-1* since each level contains all the data in the previous level in addition to new data.

As we can see from Figure 7, the exclusion effect for small values of r results mainly from filter one. We can also see that, in general, the exclusion power of filter one decreases as the resolution level gets higher. The exclusion power of filter one also decreases when r gets larger as we can see from Figure 8 which shows the exclusion process of the same datasets presented in Figure 7 but for values of r that return 10% of the time series

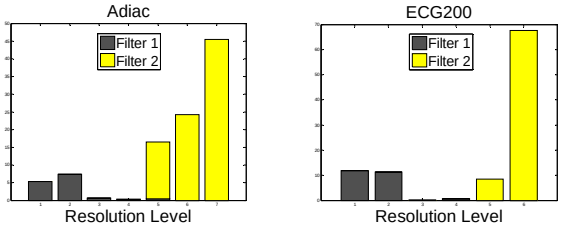

Figure 8: The exclusion process of datasets (Adiac) (left) and (ECG200)(right) for *r* that returns 10% of the time series.

5. DISCUSSION

The experiments show that the performance of the two filters seems to be complementary, since most of the time series at lower resolution levels are excluded by filter one, while most of the time series at higher resolution levels are excluded by filter two . This phenomenon can be explained by examining relations (7) and (9): at lower resolution levels the segments are longer, so the approximation error is higher. As a consequence, the absolute difference in filter one is a difference between relatively large numbers: $d(Q,\widetilde{Q}^{(k)})$, $d(S,\widetilde{S}^{(k)})$,so this difference has a better chance of exceeding r and excluding the time series than at higher levels, where the approximation is better, so these numbers become smaller and their difference has less chance of exceeding r

The performance of filter two is different; at lower levels $d^R(S^{R(k)},Q^{R(k)})$ is small while $d(Q,\widetilde{Q}^{(k)})+d(S,\widetilde{S}^{(k)})$ is relatively large (see the beginning of this section), so the chance for this filter to exclude time series is low. As the resolution level gets higher $d^R(S^{R(k)},Q^{R(k)})$ gets bigger, while $d(Q,\widetilde{Q}^{(k)})+d(S,\widetilde{S}^{(k)})$ gets smaller, so this filter has a better chance of excluding time series at higher levels

An interesting phenomenon we noticed is that in some datasets, for very small r, the performance of MIR drops as we use a higher degree approximating function. We think the reason for this is that the algorithm pays an overhead cost when using a higher degree approximation.

We also notice that as the degree of the approximating function gets higher, the performance of the first filter deteriorates.

In general, the exclusion process of the proposed method is complex since at each step the number of time series that can be excluded depends on what has been excluded so far. As we mentioned in section 4, this depends on the dataset, the resolution level and the value of r

6. CONCLUSION AND FUTURE WORK

In this paper we presented a new method that uses a different paradigm to tackle the similarity search problem. The conducted experiments using the proposed method give promising results. In all the experiments, the performance was much better than that of sequential scanning for small values of r, and was better than sequential scanning, even for large values of r .

In this paper we presented the results obtained by using the Euclidean distance, but our method can support a variety of distances. We conducted several preliminary experiments using other distances and they gave similar results.

While working on this method and conducting the experiments we realized that there are many heuristics that can be used to improve it. The first direction of improvement is to optimize the filtering process by sorting the distances before filtering. The preliminary experiments that we conducted using some optimising heuristics showed that sorting improved the performance of our method. Other directions of optimizing, like recycling some calculations from lower levels, gave good results too, yet we need to find the best recycling scheme.

The approximating functions we used in this paper were polynomials, but we think that other types of functions, which are particularly designed to approximate time series, may even give better results.

The fact that the exclusion power depends on several factors can be exploited by training the algorithm to find the best resolution levels to be used for a certain dataset and for a particular value of r . This can improve the performance of our method.

Another, and more important, direction of future work is to use one of the dimensionality reduction methods, which are well-known in the literature, as an approximating function. The principle here is that dimensionality reduction methods aim at finding the best representation of the original space at a lower space, so this can be viewed as an approximation of the time series.

When we started developing our method, our aim was also to use it on multidimensional time series, so this is still a main direction of future work.

The final direction of future work is to apply our method to other data types where the idea of resolution level is pertinent.

7. REFERENCES

[1] Agrawal, R., Faloutsos, C., & Swami, A. 1993. Efficient similarity search in sequence databases". Proceedings of the 4th Conf. on Foundations of Data Organization and Algorithms.

[2] Agrawal, R., Lin, K. I., Sawhney, H. S. and Shim, K. 1995. Fast similarity search in the presence of noise, scaling, And translation in time-series databases, in Proceedings of the 21st Int'l Conference on Very Large Database s. Zurich, Switzerland, pp. 490-501.

[3] Cai, Y. and Ng, R. 2004.Indexing Spatio-temporal Trajectories with Chebyshev Polynomials. In SIGMOD.

[4] Chan, K. & Fu, A. W.1999. Efficient Time Series Matching by Wavelets. In proc. of the 15th IEEE Int'l Conf. on Data Engineering. Sydney, Australia, Mar 23-26.

[5] Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra. 2000. Dimensionality reduction for fast similarity search in large time series databases. J. of Know. and Inform. Sys.

[6] Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra.2001. Locally adaptive dimensionality reduction for similarity search in large time series databases. SIGMOD, pp 151-162

[7] Korn, F., Jagadish, H & Faloutsos. C.1997. Efficiently supporting ad hoc queries in large datasets of time sequences. Proceedings of SIGMOD '97, Tucson, AZ, pp 289-300

[8] Marteau, PF. 2009. Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching. IEEE Trans. Pattern Anal. Mach. Intell. 31(2): 306-318

[9] Morinaka, Y., Yoshikawa, M. , Amagasa, T., and Uemura, S.2001. The L-index: An indexing structure for efficient subsequence matching in time sequence databases. In Proc. 5th PacificAisa Conf. on Knowledge Discovery and Data Mining, pp 51-60

[10] Schulte,M.J. ,Lindberg, M. and Laxminarain, 2005. A. Performance Evaluation of Decimal Floating-point Arithmetic in IBM Austin Center for Advanced Studies Conference, February 2005.

[11] Yi, B,K., & Faloutsos, C.2000. Fast time sequence indexing for arbitrary Lp norms. Proceedings of the 26st International Conference on Very Large Databases, Cairo, Egypt,

[12] UCR Time Series Classification/Clustering Page http://www.cs.ucr.edu/~eamonn/time_series_data/