Fast Retrieval of Time Series Using a Multi-resolution Filter with Multiple Reduced Spaces

Muhammad Marwan Muhammad Fuad, Pierre-François Marteau

Fast Retrieval of Time Series Using a Multi-resolution Filter with Multiple Reduced Spaces

Muhammad Marwan Muhammad Fuad[1], Pierre-François Marteau[1]

[1]VALORIA, Université de Bretagne Sud, Université Européenne de Bretagne
BP. 573, 56017 Vannes, France
{marwan.fuad, pierre-francois.marteau}@univ-ubs.fr

Abstract: Fast retrieval of time series that are similar to a given pattern in large databases is a problem which has received a lot of attention in the last decade. The high dimensionality and large size of time series databases make sequential scanning inefficient to handle the similarity search problem. Several dimensionality reduction techniques have been proposed to reduce the complexity of the similarity search. Multi-resolution techniques are methods that speed-up the similarity search problem by economizing distance computations. In this paper we revisit two of previously proposed methods and present an improved algorithm that combine the advantages of these two methods. We conduct extensive experiments that show the show the superior performance of the improved algorithm over the previously proposed techniques.

Keywords: Time Series Databases, Similarity Search, Dimensionality Reduction Techniques, Sequential Scanning, MIR, MIR_X, Tight-MIR.

1 Introduction

Time series similarity search is a problem that has many applications in computer science. Similarity between two time series can be depicted using a similarity distance, which is usually costly compared with other tasks such as CPU time or even I/O time.

Formally, range query can be defined as follows: given a time series database U of size n and a query (q, r), where r represents a *threshold.* The range query problem can be specified as retrieving all the time series $u \in U$ which satisfy: $d(q, u) \leq r$.

The trivial answer to this problem that sequential scanning offers, which is based on scanning U and returning the time series that satisfy the above condition, is slow because it requires n distance computations, which can be computationally very expensive in large databases with costly distances.

Time series dimensionality reduction techniques aim at reducing this high complexity by transforming the time series into a lower dimensional space, thus decreasing query-time distance evaluations, and processing the similarity search in this reduced space.

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) [10], *Adaptive Piecewise Constant Approximation* (APCA) [9], *Piecewise Aggregate Approximation* (PAA) [8,18], *Piecewise Linear Approximation* (PLA) [11], *Chebyshev Polynomials* (CP) [3]

Time series dimensionality reduction techniques are based on predefined schemes, in that the parameters which control their performance, and which have been chosen at indexing-time, may prove to be inappropriate at query-time. Multi-resolution techniques offer more control on the parameters that determine the effectiveness and efficiency of dimensionality reduction methods. In [12] and [13] two such multi-resolution techniques have been proposed. In this paper we propose an improved algorithm which the advantages of the two previously proposed techniques in terms of performance and autonomy.

The work presented in this paper is organized as follows: in section 2 we present related work and background. In section 3 of this paper we introduce the improved algorithm and we evaluate its performance in section 4. In section 5 we give concluding remarks

2 Related Work and Background

2.1 Similarity Distances

Let D be a set of objects. A function $d : D \times D \to \Re^+ \cup \{0\}$, is called a distance metric if the following holds:

(p1) $d(x, y) \geq 0$ (non-negativity)

(p2) $d(x, y) = d(y, x)$ (symmetry)

(p3) $x = y \Leftrightarrow d(x, y) = 0$ (identity)

(p4) $d(x, z) \leq d(x, y) + d(y, z)$ (triangular inequality)

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

2.2 Dimensionality Reduction Techniques

There are several dimensionality techniques in the literature, we present in the following two of them

Piecewise Aggregate Approximation (PAA): This method was proposed in [8] and [18], independently. Its basis is simple and straightforward, yet this method has been successfully used as a competitive method.

PAA reduces the dimensionality of a time series from *n* in the original space to *N* in the reduced space by segmenting the time series into equal-sized frames and representing each segment by the mean of the data points that lie within that frame.

The similarity distance used in the reduced space is:

$$d(X,Y) = \sqrt{\frac{n}{N}}\sqrt{\sum_{i=1}^{N}(\overline{x_i} - \overline{y_i})^2} \quad (1)$$

Where *n* is the length of the time series, N is the number of frames, which should be a factor of n. The compression ratio m is n/N.

It is proven in [8] and [18] that the above similarity distance is lower bounding of the Euclidean distance applied in the original space of time series.

Since d^N is lower bounding of the Euclidean distance in the original space then, according to the GEMINI algorithm, which is a generic approach of indexing and retrieving time series [6], any time series u that satisfies:

$$d^N(q,u) > r \quad (2)$$

can not be answer to the query and should be excluded.

The Symbolic Aggregate Approximation (SAX): Symbolic representation of time series has attracted much attention recently, because by using this method we can not only reduce the dimensionality of time series, but also benefit from the numerous algorithms used in bioinformatics and text data mining. Of all the symbolic representation methods, the symbolic aggregate approximation method (SAX) [7] is one of the most powerful ones in time series data mining.

SAX is based on the fact that normalized time series have highly Gaussian distribution [7], so by determining the breakpoints that correspond to the chosen alphabet size, one can obtain equal sized areas under the Gaussian curve.

SAX is applied in the following steps: in the first step all the time series in the database are normalized. In the second step, the dimensionality of the time series is reduced by using PAA. In the third step, the PAA representation of the time series is discretized. This is achieved by determining the number and the locations of the breakpoints. This number is related to the chosen alphabet size (it is chosen by the user), i.e. *alphabet_size=number(breakpoints)+1* . The locations of the breakpoints are determined by statistical lookup tables so that these breakpoints produce equal-sized areas under the Gaussian curve. The interval between two successive breakpoints is assigned to a symbol of the alphabet, and each segment of the PAA that lies within that interval is discretized by that symbol.

The last step of SAX is using the following similarity distance:

$$MINDIST(\hat{S},\hat{T}) \equiv \sqrt{\frac{n}{N}}\sqrt{\sum_{i=1}^{N}(dist(\hat{s}_i,\ \hat{t}_i))^2} \quad (3)$$

Where n is the length of the original time series, N is the number of the frames, $\tilde{S}$ and $\tilde{T}$ are the symbolic representations of the two time series S and T, respectively, and where the function $dist(\ )$ is implemented by using the appropriate lookup table.

3 The Proposed Algorithm

A Multi-resolution Indexing and Retrieval scheme (MIR) has been proposed in [12]. This scheme is based on using several reduced spaces that correspond to different resolution levels. The principal of the algorithm is as follows: let U 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 $u \in U$ is divided into m segments. Each segment is approximated by a function of low dimension. The functions used are polynomials of degree 1, 3, or 5, and where the approximation error, according to a given distance, between this segment and the approximating function is minimal, so this function is the best approximation of that segment. A function of the same type and the same degree is used to approximate all the segments of all the time series in the database. The image vector $\bar{u}$ is, by definition, an n-dimensional vector whose components are the images of all the points of all the segments of a time series on that approximating function. The images of the two end points of that segment on the approximating function are called 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 u^R of the time series on R. Two distances are defined on the database: the first is denoted by d, and is defined on a n-dimensional space, so it is the distance between two time series in the original space , i.e. $d(u_i, u_j)$, or the distance between the original time series and its image vector, i.e. $d(u_i, \bar{u}_i)$. 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. $d^R(u_i^R, u_j^R)$ .It is shown in [12] that d^R is a lower bounding of d when the Minkowski distance is used. 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 further segmentation of the time series using different values $m \le m_k$, The image vector and the projection vector at level k are denoted by $\bar{u}^{(k)}$ and $u^{R(k)}$, respectively.

Given a query (q, r), let $\bar{u}$, $\bar{q}$ be the projection vectors of u, q, respectively, on their approximating functions, where u is a time series in the database. By applying the triangular inequality we get:

$$\left|d(u,\bar{u}) - d(q,\bar{q})\right| > r \quad (4)$$

This relation represents a pruning condition which is called the first filter.

By applying the triangular inequality again we get :

$$d^R(u^{R(k)}, q^{R(k)}) > r + d(q,\bar{q}^{(k)}) + d(u,\bar{u}^{(k)}) \quad (5)$$

This relation represents the second filter.

At indexing-time the application of the method starts by choosing the length of segments for each resolution level. The shortest length corresponds to the lowest level, and the longest corresponds to the highest level. Then the approximating function to be used with all the time series and for all resolution levels is chosen. The distances $d(u,\bar{u}^{(k)}) \quad \forall u \in U$ are then computed and stored. The segmenting scheme is simple and straightforward. The segments are connected and their length is a power of 2. So for a time series of 152 dimensions, for instance, the segmenting scheme for the first resolution level is: $[1,64],[64,128],[128,152]$. Notice that the three segments have three different lengths (63,64,14), respectively, and this difference of lengths will continue to appear at higher levels. Notice also that this segmentation contains the information of four points (1,64,128,152).

At query-time the query is divided into segments with the same length as that of the time series and for each resolution level. These segments are approximated using the same approximating function that was used to approximate the time series. The distances $d(q,\bar{q}^{(k)})$ are then computed. The algorithm tries to exclude the time series at each resolution level using the first filter which is less costly to apply than the second filter, because it does not include any distance evaluation at query-time. The second filter uses two distances that have been computed at indexing-time: $d(q,\bar{q}^{(k)}), d(u,\bar{u}^{(k)})$, so the only distance that is to be computed at query-time is $d^R(u^{R(k)}, q^{R(k)})$. The second filter is less costly at lower levels than at higher levels. But at any level, the cost of applying the second filter is never as costly as the distance computations at the original space, because it is assumed that $2m \le n$.

MIR starts with the lowest level and tries to exclude the first time series using (4). If this time series is pruned, the algorithm moves to the next time series, if not, the algorithm tries to prune this time series using relation (5). If all the time series in the database have been pruned 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 all the false alarms and return the true answer set.

In [13] a multi-resolution scheme is presented. This scheme combines the first filter as described in (4) with the exclusion condition of one of the dimensionality reduction techniques known in the literature. This algorithm is called MIR_X, where X is the dimensionality reduction technique used. In MIR_X each segmented time series has two representations; one that is identical to that of MIR and the other is the

representation proposed by the dimensionality reduction technique used. The algorithm used is similar to that of MIR. The experiments show that MIR_X improves the performance of dimensionality reduction techniques

Both MIR and MIR_X have their drawbacks: The two distances $d(q,\overline{q}^{(k)}), d(u,\overline{u}^{(k)})$ used to apply the second filter (relation (5)) lower its pruning power. Besides, the segmenting scheme used is not optimal and can results in segments of completely different lengths when the length of the time series is not a power of 2. The direct consequence of this segmentation is that the approximation error is not harmonized which, in turn, lowers the pruning power of the first filter. MIR has one main advantage; it is a standalone method, unlike MIR_X

MIR_X has the same segmenting scheme as that of MIR. However, using the exclusion condition of a dimensionality reduction technique which , in the case of the dimensionality reduction techniques used in [13], uses a lower bounding condition to the distance in the original space makes the second filter described in (5) redundant, because it is overwritten by the more powerful exclusion condition of the dimensionality reduction technique. Hence, the performance of MIR_X is better than that of MIR. Nevertheless, MIR_X has a few drawbacks: it is completely dependent on the dimensionality reduction technique used. If the dimensionality reduction technique uses a lower bounding condition, MIR_X can be applied, if not, MIR_X can never be applied. On the other hand, the application of MIR_X requires adopting a different concept of resolution level for each dimensionality reduction technique, which is not intuitive. Besides, some dimensionality reduction techniques have certain requirements (the length of the time series should be a power of 2 for DWT, N should be a factor of n for PAA and SAX). All these factors influence the application of MIR_X.

In this paper we present an improvement on the two previous versions. This improved algorithm, which we call *Tight-MIR*, has the advantages of both MIR and MIR_X in that it is a standalone method, yet it has the same competitive performance of MIR_X.

The redundancy of the second filter in MIR_X suggests that the application of MIR can use two separate filters. In fact the requirement that the approximating error of the approximation function be minimal is not exploited when constructing the second filter as described in [12].

In Tight-MIR instead of using the projection vector to construct the second filter, we access the raw data in the original space directly using a number of points that corresponds to the dimensionality of the reduced space at that resolution level. In other words, we use $2m$ raw points, instead of $2m$ main images, to compute d^R. There are several positive effects to this modification; the first is that the new d^R is obviously tighter than d^R as computed in [12]. The second is that when using a Minkowski distance d^R is lower bounding to the original distance in the original space. The direct consequence of this is that the two distances $d(q,\overline{q}^{(k)}), d(u,\overline{u}^{(k)})$ become redundant, so the second filter is overwritten by the usual lower bounding condition $d^R(u^{R(k)}, q^{R(k)}) > r$.

Notice that the complexity of the modified d^R is $O(2m)$ which is the same complexity as described in [12]. So this modification does not require any extra cost

Tight-MIR also utilizes an improved segmenting scheme: at each level, the segments are equal and disconnected, so for a time series of 152 dimensions (the one presented earlier this section) the segmentation used for the first level is $[1,76],[77,152]$, this segmentation has two advantages: the error is more harmonized . The second advantage is that although we are using two segments only (at the first resolution levels) we are representing the information of four points of four points (1,76,77,152) , which is the same number of points presented in the original segmenting. So the modified algorithm needs one more resolution level less than the original segmenting, which is another advantage in terms of space complexity.

4 Performance Evaluation

The objective of our experiments is to show that the modified algorithm Tight-MIR has the advantages of both MIR and MIR_X together, so we have to show that it outperforms MIR, and that it has the same performance as that of MIR_X. We also conduct other experiments to compare Tight-MIR against other dimensionality reduction techniques, because Tight-MIR it is a standalone method (unlike MIR_X).

Although several papers present experiments based on wall clock time, but it is a poor choice and subject to bias [5,8], so we prefer to use the latency time concept presented in [15]. Latency time refers to the number of cycles the processor takes to perform different arithmetic operations (>,+ - ,*,abs, sqrt) when performing the similarity search. Then the number of each operation is multiplied by the latency time of that operation to get the total latency time of the similarity search. The latency time is 5 cycles for (>, + -), 1 cycle for (abs), 24 cycles for (*), and 209 cycles for (sqrt). This method of testing performance is the same one that was used in both [12] and [13]

We conducted extensive experiments using datasets of different sizes and dimensions and from different repositories [14, 16, 17, 19]. We also used different threshold values. We conducted experiments using approximating polynomials of different degrees, but because of space limitation we report here the results of approximating by first degree polynomials to facilitate the comparison.

We first show a comparison between MIR and Tight-MIR. The way d^R is computed in Tight-MIR enables us to modify the codes used to avoid the square root, which is a very costly operation, of course sequential scanning was also modified to avoid this operation. This modification is not possible with MIR. So the comparison we present here is made between the speed-up of MIR and Tight-MIR compared to sequential scanning. In Figure 1 we present the results of four datasets. The results clearly show that Tight-Mir outperforms MIR for all the datasets and for all values of r . As in the experiments of [12] and [13], the values of r vary between those which return 1% and 10% of the time series in sequential scanning.

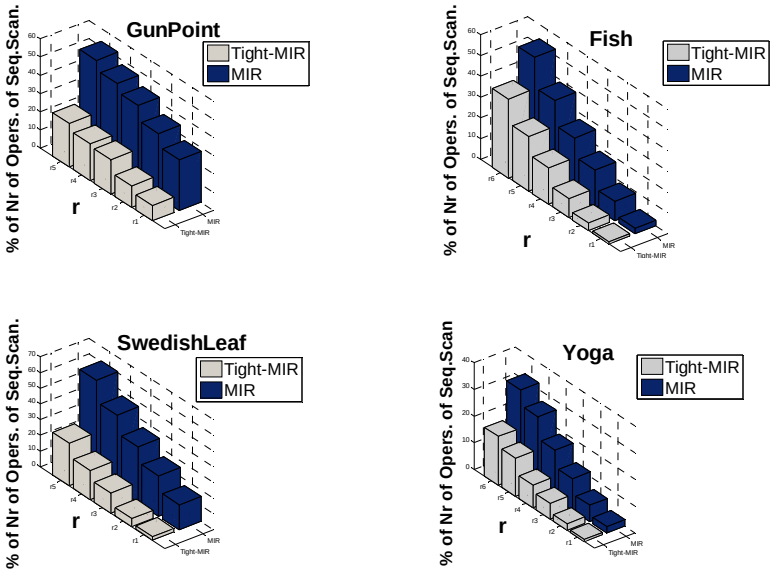

Fig. 1. Comparison of the speed-up of MIR and Tight-MIR on four datasets (Yoga), (SwedishLeaf), (GunPoint) and (Fish) over sequential scanning.

In the second series of experiments we compared MIR_X, where X is dimensionality reduction technique, with Tight-MIR to show that the two methods give similar results. Figure 2 shows some of the results we obtained comparing

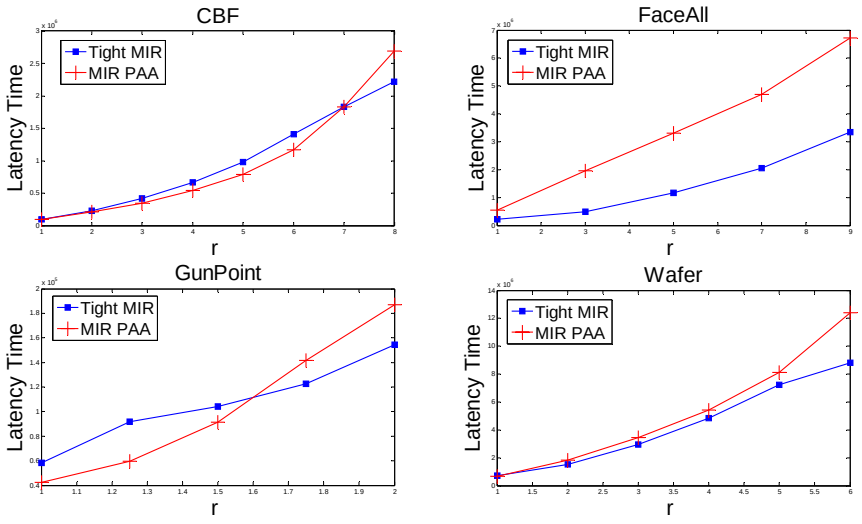

Fig. 2. Comparison between MIR_PAA and Tight-MIR on datasets (CBF), (FaceAll), (Wafer), and (GunPoint)

Tight-MIR with MIR_PAA. The results presented in Figure 2 show that MIR_PAA and Tight-MIR have the same performance. In fact, we can even say that the performance of Tight-MIR is even slightly better.

Comparing Tight-MIR with MIR_SAX also showed that the two methods have the same performance.

It is important to mention that both MIR_PAA and MIR-SAX require that N be a factor of n, the length of the time series (but Tight-MIR does not require that).

We also conducted other experiments to compare Tight-MIR with PAA, using different datasets and different values of r. We report in Figure 3 some of the results we obtained. The results show that Tight-MIR outperforms PAA for all the datasets

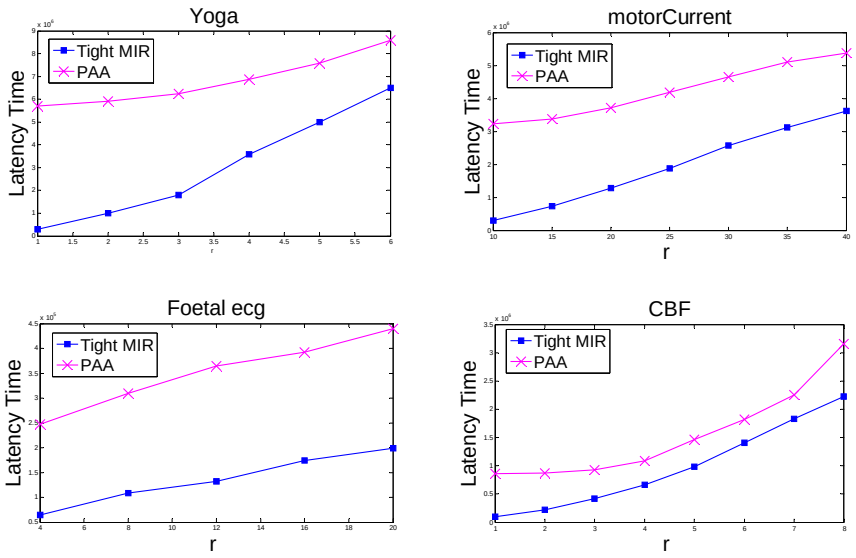

Fig. 3. Comparison between PAA and Tight-MIR on datasets (Yoga), (motorCurrent), (CBF), and (Foetal ecg)

We also conducted experiments comparing Tight-MIR with SAX, which is a very competitive and fast method. SAX appeared in two versions; in the first one the alphabet size varied in the interval (3:10), and in the second one the alphabet size varied in the interval (3:20).

We conducted experiments on different datasets, and for different values of the alphabet size. The codes we used in the experiments were optimized versions of the original codes of SAX, since the original codes written by the authors of SAX were not optimized for speed, so we optimized them to make a fair comparison.

We report in Figure 4 the results of several datasets and for different values of r. The results shown here are for alphabet size 3 (the smallest alphabet size possible for SAX), 10 (the largest alphabet size in the first version of SAX), and 20 (the largest alphabet size in the second version of SAX).

We have to mention that the datasets used in these last experiments were normalized because SAX can only be applied to normalized time series.

The results obtained show that Tight-MIR clearly outperforms SAX for the different values of r and for the different values of the alphabet size.

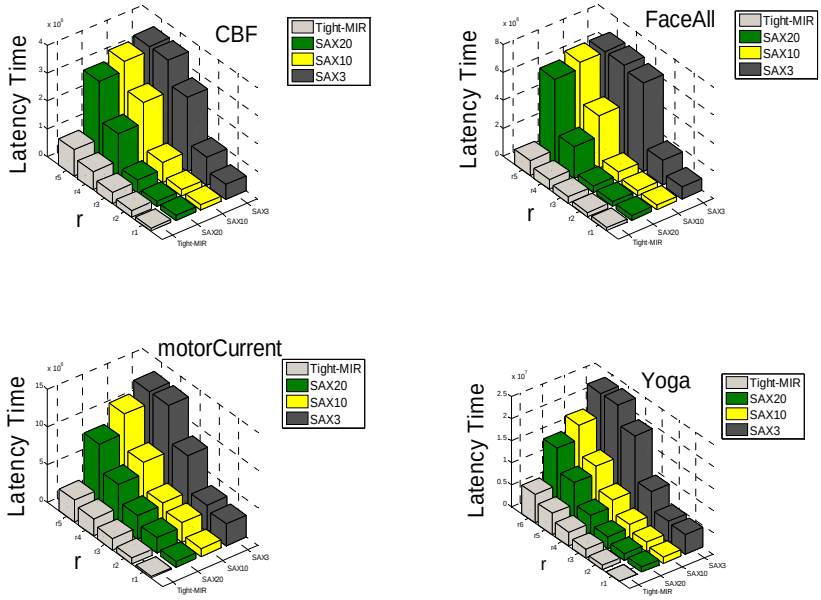

Fig. 4. Comparison of the latency time between Tight-MIR and SAX for alphabet size=3, 10, and 20 on datasets (FaceAll), (CBF),(motoCurrent), and (Yoga)

It is important to mention that the results of SAX as show in the above Figure may give the fake impression that with some datasets the number of operations seems to be stable after a certain value of r. This phenomenon does not indicate stability of performance. It only indicates that the SAX exploited all the indexed time series using the lower bounding condition without being able to exclude any time series, so the search process moved to sequential scanning. So this phenomenon is the worst scenario possible because the number of operations exceeds even that of sequential scanning and reaches the maximum number possible of operations, i.e. the maximum number of distance evaluations.

In all the experiments we presented so far comparison was made based on speed as measured by the latency time, but comparisons between representation methods can also be made according to their pruning power. In Table 1 we present the pruning power of MIR-Tight and SAX with alphabet size=20 (which is the most effective version of SAX) on the datasets presented in Figure 4. The results show that the pruning power of Tight-MIR is much more stable than that of SAX20 as r gets larger.

Table 1. Comparison of the pruning power between Tight-MIR and SAX20 (alphabet size=20) for the smallest and largest values of r that were used in the experiments presented in Figure 4.

	SAX20		Tight-MIR	
	r_{min}	r_{max}	r_{min}	r_{max}
CBF	99.89 %	14.88 %	99.89 %	98.69 %
FaceAll	99.94 %	7.51 %	99.94 %	98.89 %
motoCurrent	98.40 %	33.15 %	99.87 %	88.13 %
Yoga	99.53 %	37.61 %	99.63 %	85.84 %

In the final set of experiments we wanted to test if the performance of Tight-MIR is stable with different dimensions of time series. We conducted experiments using datasets of different dimensions and the results showed high stability of performance. We present in Figure 5 the results of applying Tight-MIR on dataset (Wind) whose dimension is 12, and dataset (motoCurrent) whose dimension is 1500, compared to sequential scanning which represents the baseline performance.

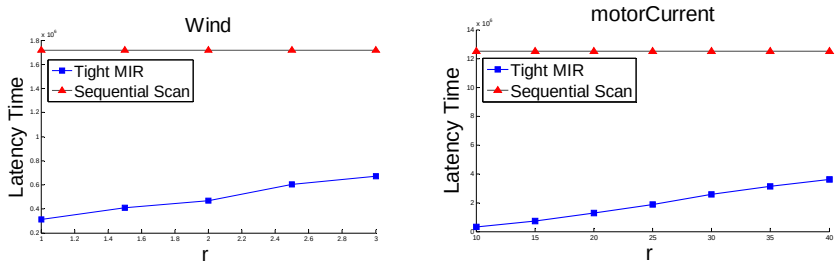

Fig. 5. Comparison of the latency time between Tight-MIR and Sequential Scanning on datasets (Wafer) and (motoCurrent)

5 Conclusion

In this work we presented an improved multi-resolution algorithm of time series retrieval. This new algorithm combines the advantages of two previously proposed algorithms. We conducted extensive experiments comparing the new algorithm with the previously proposed algorithms. The results of the experiments show the superiority of the improved algorithm over the two previous ones.

We also conducted other experiments which compare the performance of the new algorithm with other dimensionality reduction techniques. The results also show that the improved algorithm outperforms those tested dimensionality reduction techniques both in terms of speed and pruning power.

References

1. Agrawal, R., Faloutsos, C., & Swami, A.: Efficient Similarity Search in Sequence Databases". Proceedings of the 4th Conf. on Foundations of Data Organization and Algorithms (1993)
2. Agrawal, R., Lin, K. I., Sawhney, H. S. and Shim, K.: 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 Databases. Zurich, Switzerland, pp. 490-501(1995).
3. Cai, Y. and Ng, R. : Indexing Spatio-temporal Trajectories with Chebyshev Polynomials. In SIGMOD (2004).
4. Chan, K. & Fu, A. W.: Efficient Time Series Matching by Wavelets. In proc. of the 15th IEEE Int'l Conf. on Data Engineering. Sydney, Australia, Mar 23-26. pp 126-133 (1999).
5. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., and Keogh, E.: Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. In Proc of the 34th VLDB (2008).
6. Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. : Fast Subsequence Matching in Time-series Databases. In Proc. ACM SIGMOD Conf., Minneapolis (1994)
7. Jessica Lin, Eamonn J. Keogh, Stefano Lonardi, Bill Yuan-chi Chiu: A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. DMKD (2003).
8. Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra: Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. J. of Know. and Inform. Sys. (2000).
9. Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra: Locally Adaptive Dimensionality Reduction for Similarity Search in Large Time Series Databases. SIGMOD (2001).
10. Korn, F., Jagadish, H & Faloutsos. C.: Efficiently Supporting ad hoc Queries in Large Datasets of Time Sequences. Proceedings of SIGMOD '97, Tucson, AZ,pp 289-300 (1997).
11. Morinaka, Y., Yoshikawa, M. , Amagasa, T., and Uemura, S.: 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, pages 51-60 (2001).
12. Muhammad.Fuad, M.M., Marteau P.F., "Multi-resolution Approach to Time Series Retrieval", Fourteenth International Database Engineering & Applications Symposium– IDEAS 2010 , Montreal, QC, Canada (2010)
13. Muhammad.Fuad, M.M. , Marteau P.F.," Speeding-up the Similarity Search in Time Series Databases by Coupling Dimensionality Reduction Techniques with a Fast-and-dirty Filter ", Fourth IEEE International Conference on Semantic Computing– ICSC 2010,Carnegie Mellon University, Pittsburgh, PA, USA (2010)
14. http://povinelli.eece.mu.edu/
15. Schulte,M.J. ,Lindberg, M. and Laxminarain, A. :Performance Evaluation of Decimal Floating-point Arithmetic in IBM Austin Center for Advanced Studies Conference, February (2005).
16. SISTA's Identification Database http://www.esat.kuleuven.ac.be/~tokka/daisydata.html
17. StatLib - Datasets Archive http://lib.stat.cmu.edu/datasets/
18. Yi, B,K., & Faloutsos, C.: Fast Time Sequence Indexing for Arbitrary Lp norms. Proceedings of the 26st International Conference on Very Large Databases, Cairo, Egypt (2000).
19. UCR Time Series datasets http://www.cs.ucr.edu/~eamonn/time_series_data/